Vysoká škola báňská – Technická univerzita

Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

PHP Frameworky

PHP Frameworks

2010 **Jakub Vaněk**

VŠB – Technická univerzita Ostrava Fakulta elektrotechniky a informatiky

Katedra informatiky Akademický rok 2009/2010

Zadání bakalářské práce

Řešitel: **Jakub Vaněk**

Obor: 2612R025 Informatika a výpočetní technika

Téma: PHP Frameworky

Cílem bakalářské práce je popsat práci a využití PHP Frameworků a navrhnout rozšíření vybraných frameworků

1. Popište význam, použití, výhody a nevýhody frameworků.
2. Popište technologii PHP a PHP Frameworků.
3. Uveďte a vyberte několik zástupců z PHP Frameworků, pro vybrané PHP Frameworky proveďte jejich důkladný popis a porovnání.
4. Popište možnosti rozšíření vybraných frameworků.
5. Rozšiřte alespoň jeden PHP Framework o vybranou vlastnost.
6. Znázorněte a popište použití frameworků v praxi.

Odborná literatura: Podle pokynů vedoucího

Vedoucí: **Ing. Tomáš Drábek**

Datum zadání: 20. listopadu 2009

Termín odevzdání: 7. května 2010

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením vedoucího práce. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: …… Podpis: ……...

Poděkování

Tímto děkuji vedoucímu bakalářské práce Ing. Tomášovi Drábkovi za obětavost, preciznost, odbornou pomoc a hlavně ochotu pomoci a poradit vždy, když při zpracování mé bakalářské práce bylo potřeba.

Abstrakt

Tato práce má za úkol poskytnout čtenáři obecný rozhled o vlastnostech, možnostech využití a důvodech, proč právě při vývoji webových aplikací v programovacím jazyce PHP využít PHP Frameworků. Jde o moderní trend vývoje aplikací, kdy dochází k značnému zjednodušení vývoje a k neméně důležité úspoře času. V této práci je možné naleznou seznámení s důležitými pojmy, díky kterým získá čtenář základy a prvotní přehled při studiu tohoto zajímavého tématu. V úvodu je možné nalézt důvody proč zvolit právě PHP frameworky a jaké aspekty hrají důležitou roli při jejich výběru.

Dále je práce zaměřená na popis představitelů z řad frameworku. Díky tomu může čtenář získat jistý přehled o možnostech a využití. Následně byly vybrány a podrobně popsány Prado framework a Nette framework, které je možné také díky této práci osobně vyzkoušet. Čtenář zde nalezne průvodce od požadavku frameworku až po prvotní implementaci aplikace.

Na závěr toho popisu je možné vždy nalézt jejich zhodnocení. Tímto se čtenář dozví v čem daný framework vyniká a naopak v čem jsou jeho nevýhody. Konec zhodnocení jsem obohatil o mé osobní zkušenosti a doporučení, což může také pomoci při výběru frameworku.

Klíčové slova

PHP, Framework, MVC architektura, webová aplikace, Nette, Prado, komponenta

Abstract

The aim of the thesis is to provide a broad survey about features, possibilities of using and reason why to use PHP Frameworks in the web application development in PHP programming language. It is a modern trend of application development leading to substantial simplification of development and also very important time saving. The thesis includes an introduction with important terms which bring basis and very fist view to this interesting theme. The beginning of the thesis is focused on the reasons why to choose just PHP frameworks and which aspects should be taken into consideration in the process of choosing them.

Another part of the thesis is focused on the framework products description. Thanks to this a reader could be familiar with possibilities and their using. Prado Framework and Nette framework were chosen and described into details. Thanks to this thesis it is possible to try them out. Also a guide involving a framework request up to the initial implementation of application is presented here.

At the end of the description their evaluation is given. It displays the advantages and disadvantages of the particular framework. The conclusion is enriching about personal experience and recommendations that could be useful in case of framework choice.

Keywords

PHP, Framework, MVC architecture, web application, Nette, Prado, component

Seznam použitých symbolů a zkratek

ACL – Access control lists

AJAX – Asynchronous JavaScript and XML

API - Application Programming Interface

BSD - Berkeley Software Distribution

CGI - Common Gateway Interface

CRUD – Create, Read, Update, Delete

CMS - Content management systém

CSRF - Cross-site request forgery

HTML – HyperText markup language

JSON - JavaScript Object Notation

MySQL - My Structured Query Language

MVC – Model View Controller

ODBC – Open database connectivity

ORM - Object-relation mapping

PDF - Portable document format

PDO - PHP Data Objects

PHP – Personal home page

PHP/FI - Personal Home Page a Form Interpreter

SEO - Search Engine Optimization

SOAP - Simple Object Access Protocol

URL - Uniform Resource Locator

WML - Wireless Markup Language

WWW - World wide web

XHTML - Extensible hypertext markup language

XML - Extensible markup language

XSS - Cross-site scripting

OBSAH

1 ÚVOD

Dnes v moderní době, kdy přístup do sítě internetu je téměř samozřejmostí, dochází k nárůstu vzniků nových webových aplikací. Tyto aplikace pak plní nejrůznější funkce a jsou napsány různými programovacími jazyky. Tato práce se však zaměří pouze na ty, jenž jsou napsány v jazyce PHP. Tento jazyk je na trhu již řadu let a za tu dobu se stal poměrně populární. Jeho výhodou je jednoduchost a volná licence. Ovšem při implementaci brzy dojdete k zjištění, že kód je téměř jednoúčelový a hodí se vždy pro danou aplikaci. Také je mnohdy spletitý a nepřehledný či opakující se. To vše vede ke vznikům problému. Ty jsou pak nežádoucí a vedou také k růstů finančních nákladu. Existuje však moderní řešení, kdy za použití jednoduché softwarové struktury dojde k odbourání či alespoň potlačení většiny problému, které vznikají při použiti jazyka PHP.

A zde se dostáváme k problematice této práce. Ta je zaměřená na dnes velmi moderní a populární PHP Frameworky. Ty se snaží poskytnou programátorovi nový efektnější způsob řešení implementace. Frameworky obsahují velké množství knihoven, které obstarávají mnoho potřeb nutných pro ideální běh systému či aplikace. Jsou také vysoce modulární, což dovolí rozšířit Framework o novou vlastní funkci. To vše nám umožní nejen jednodušší, ale také rychlejší a přehlednější vývoj aplikace. A při práci na větším projektu usnadní práci v týmu.

Jakmile získáte prvotní znalosti, přejdete v práci k seznámení s jednotlivými představiteli. Těch existuje celá řada. Tím je zcela nemožné je všechny představit. A tak v této práci naleznete pouze hlavní představitele a získáte obecný rozhled o jejich možnostech. Tím však práce nekončí. Následuje podrobný popis dvou důležitých představitelů. Zde budete mít možnost dozvědět se jaké důležité vlastnosti obsahují a jaká je práce s těmito frameworky. Závěrem práce bude ukázka tvorby vlastní komponenty, díky které můžete při vývoji rozšířit svou aplikaci o zajímavou vlastnost.

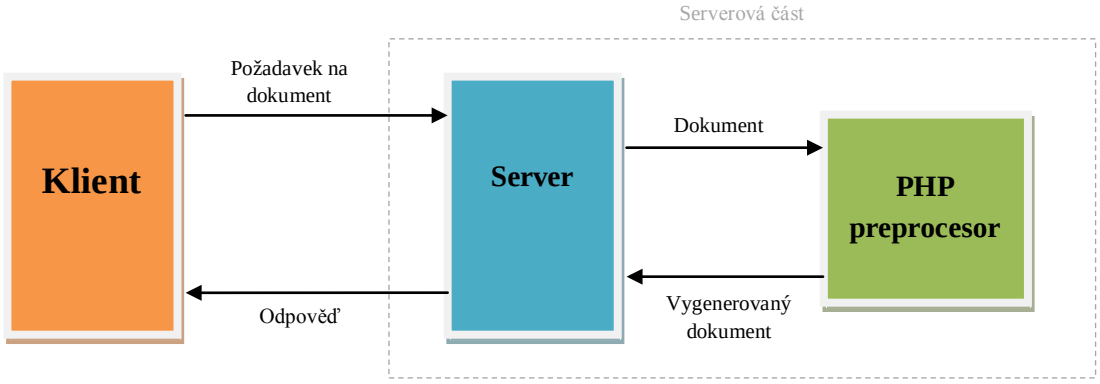
2 PHP

Jde o dnes dosti rozšířený programovací jazyk. Přesněji tedy o skriptovací programovací jazyk. Ten je především určen k tvorbě dynamických webových stránek a aplikací. Při tvorbě webových aplikací je PHP začleněno do struktury jiného jazyka, s nimž tvoří jeden celek a dotváří tak celou webovou aplikaci. Jazyky vhodné k začleňování PHP jsou například HTML, XHTML , WML apod. To však nemusí být jediné využití tohoto skriptovacího programovacího jazyka. PHP jde také využít při tvorbě konzolových či desktopovych aplikací. To ovšem není až tak běžné.

PHP jako jazyk vycházel z různých jazyků z nichž také zčásti přebral syntaxe. Proto podobnost můžeme najít s jazyky jako Perl, Java, C a Pascal. Jistou výhodou při prvním setkání může být znalost již některého z programovacích jazyků. PHP je také nezávislé na platformě. Což znamená, že jeho skripty můžou bez větších úprav fungovat na různých operačních systémech. Fungování PHP skriptu si popíšeme v následující části dokumentu

2.1 Funkce PHP

PHP funguje tak, že skript je odeslán na stranu serveru. Na této straně se skript provede. Skript se na serveru provádí pomoci PHP preprocesoru. Ten má za úkol vygenerovat dokument ze skriptu, jenž je mu poslán. Poté co je dokument vygenerován je tato odpověď nachystána na straně serveru k odeslání. Odpověď se odesílá zpět ke klientovi, u kterého je zobrazen výsledek PHP skriptu. Pro jednoduchost představení fungování PHP skriptu je možno nahlédnout na tento obrázek.



Zdroj: *Vlastní*

Obrázek 1: Princip komunikace mezi klientem a serverem

Výhoda takovéto funkce je, že veškeré generování se provádí na straně serveru. Díky tomu klient není nijak zatěžován. Předpoklad je, že čím více přístupu bude na server v jeden okamžik vyvíjeno, tím kvalitnější server musí být. Touto kvalitou mám na mysli hardwarovou výkonnost serveru.

PHP má podporu mnoha knihoven. Ty se mohou použít pro různé účely. Ať už při práci s textem, grafikou, databází (MySQL, ORACLE), či práci se soubory. Tímto se jazyk stal velmi populárním. Například spojení PHP a MySQL je velmi známé a hojně používané. PHP jako takové není vůbec těžké a i snad díky tomu nabylo na popularitě. Nebo snad také díky tomu, že používá různé vlastnosti různých programovacích jazyků. Jedno je však jisté, PHP jako jazyk si prošlo nemalou historií, jenž stojí alespoň za zmínku.

2.2 Historie PHP

PHP byla původně zkratka pro Personal Home Page[1]. Šlo tedy o osobní domácí stránky. Tento pojem se poprvé ukázal světu v roce 1994, a to díky panu Rasmusovi Lerdorfovi. Rasmus Lerdorf napsal v daném roce binární část Common Gateway Interface zkráceně CGI[1], a to v programovacím jazyku C. Tuto část napsal jako nástroje pro Personal Home Page za účelem možné záměny s malou skupinou skriptů v Perlu. Tyto skripty měly sloužit pro údržbu osobní domovské stránky. Měly například zajistit běh úloh, jako zobrazení obsahu a zaznamenávání návštěvnosti jeho stránek. Ještě tentýž rok, tedy 1994, skloubil tento kód s jiným programem, který sám napsal. Vznikla tak kombinace PHP/FI, a to spojením Personal Home Page a Form Interpreter. Tato kombinace již měla mnohem větší funkčnost. PHP/FI obsahovala širokou implementaci pro programovací jazyk C a již v této verzi se objevila komunikace s databázemi. To byl velký krok kupředu. Umožnilo to tvorbu prvních jednoduchých webových aplikací.

Oficiální představení PHP proběhlo 8. června 1995, kdy jej Rasmus Lerdorf veřejně vydal. Vydal jej hlavně z důvodu, aby mohl najít co nejvíce chyb a zdokonalit tak kód. Tato verze nesla označení PHP 2 a jednalo se o druhou verzi. Tato verze již obsahovala základní funkčnost jako dnešní PHP, tedy použití proměnných ve stylu Perlu, zpracování formulářů a možnosti vložení HTML kódu.

[1]CGI – Common Gateway Interface – jde o protokol pro propojení externích aplikací s webovým serverem[2].

K další zásadní změně došlo v roce 1997, kdy dva izraelští vývojáři Zeev Suraski a Andi Gutmans na Technion – Israel Institute of Technology přepsali parser. A tím umožnili vzniknout základům PHP 3. Tato změna nebyla jediná. Upravili také název na rekurzivní zkratku PHP = PHP: Hypertext Preprocessor. Tým těchto vývojářů vydal PHP/FI 2 v listopadu roku 1997. Tomuto vydání předcházelo měsíční testování beta verze. Po tomto vydání odstartovalo veřejné testování verze PHP 3. Testování probíhalo zhruba půl roku a v červnu roku 1998 došlo k oficiálnímu uvolnění verze PHP3. Ihned po vydání začali pánové Zeev Suraski a Andi Gutmans opětovně přepisovat jádra PHP. To vedlo k tomu, že v roce 1999 vydali Zend Engine. A díky tomu také založili v Izraeli, ve městě Ramat Gan, firmu s názvem Zend Technologies.

22. května 2000 došlo k vydání verze PHP 4. Tato verze je postavená na Zend Engine 1.0. Ve verzi 4, přesněji tedy 4.1, byly představeny super globální proměnné ($_GET, $_POST, $_SESSION, atd.). Také jedno z vylepšení ve verzi PHP 4 bylo použití bezpečnějších způsobů zpracování vstupů uživatelů. Uzavřela se tím jistá možnost využití bezpečnostních děr. Poslední update byl na verzi 4.8, a to dne 8. srpna 2008 - od této doby se PHP 4 přestalo vyvíjet a přešlo se na PHP 5.

PHP 5 vzniklo dne 13. června 2004. Je postaveno na Zend Engine 2.0. PHP 5 obsahuje novinky, jako je vylepšená podpora pro objektově orientované programování, dále například PHP Data Objects extension (ta definuje lehké a konzistentní rozhraní určené pro napojení k databázím) a mnoho dalších vylepšení. PHP 5 je nyní jedinou oficiální funkční vydanou verzí. Běží jak na platformě 32 bit tak i 64 bit, nicméně jako oficiální verze je uváděná pouze 32 bit.

Spolu s PHP 5 prochází vývojem již PHP 6, jenž má zatím datum vydání neznámé. Mělo by dojít k odstranění různých chyb, dále pak ke změnám ve funkcích, doplňcích a také k plné podpoře UNICODE[2], která doposud chyběla.

[2]Unicode – je tabulka znaků všech existujících abeced, která v současnosti obsahuje více než 100 000 znaků[3].

2.3 Výhody PHP

- jde o jazyk určený především pro vývoj webových aplikací
- PHP má rozsáhlou knihovnu funkcí, což umožní efektivnější a snadnější tvorbu kódu
- podpora mnohých databázových systémů
- nezávislost na operačním systému, tedy nezávislost na platformě
- Open source[3] licence
- dostupnost dokumentace a s tím související rozšířenost
- poněkud lehčí nastudování
- možnost obrovské podpory v podobě manuálů a návodů
- velké množství vytvořených projektů vhodných pro inspiraci

2.4 Nevýhody PHP

- jazyk PHP není nikde definován, pouze je popsána jeho implementace
- slabší prozatímní podpora UNICODE
- slabší některá bezpečnostní opatření
- ve standardní distribuci chybí ladící (debugovací) nástroj

[3]Open source – Počítačový software s otevřeným zdrojovým kódem.

3 Framework

Pod tímto pojmem si skrývá softwarová struktura, která je určena jako podpora programování, vývoj a organizaci jiných softwarových projektů[4]. Jelikož jde o všeobecnou definici pojmu framework, proto tuto definici podrobněji popíšeme. Framework jako takový je využíván ve více oblastech IT technologií. Rozličný bývá především různou strukturou, ale také obsahem podpůrných programů, knihoven a v neposlední řadě v návrhových vzorech. Každý framework má také jiný doporučený postup, jenž by se měl použít při vývoji aplikace. Úkolem frameworku je převzetí typických problémů v dané oblasti, které nastávají při vývoji. Tím se vývoj značně usnadní a jednotliví tvůrci, neboli také návrháři a vývojáři, se můžou pouze zaměřit na své zadání a sním související problémy. Na první pohled se může zdát, že použitím cizího kódů se kód značně zpomalí a stane se tak neefektivní. Je to ale přesně naopak, cizí kód je velkým obohacením. Dosti také bývá frameworkům vyčítáno, že při použití cizího kódu může tedy dojít k výhodám, jenž jsou užitečné při vývoji aplikace, ty jsou však draze vykoupeny časem, který je nutné věnovat k nastudování frameworku. I toto však není zdaleka taková pravda, jak by se mohlo zdát. Čas potřebný k nastudování frameworku může být dlouhý. Při jeho opakovaném nasazení použití ve velkých projektech dojde k výrazné úspoře času. Díky tomu se v dnešní době frameworky ve velké míře rozšiřují a nasazují při práci, tvorbě, vývoji, testování a mnoha dalších odvětvích. Jistou zápornou vlastností frameworku se stává ta, že pokud dojde k jeho odinstalování, může nastat problém spuštění některých aplikací.

Framework má také své složení. Rozdělen je na dvě hlavní části, a to tak zvané frozen spots a hot spots. Jak již názvy v překladu napovídají, jedná se o „zmražená“ a „horká místa“. Frozen spots definuje celkovou architekturu softwarové struktury, její základní komponenty a vztahy mezi nimi. Jednoduše řečeno definuje strukturu frameworku. Tyto části jsou tedy neměnné a zůstávají stejné při opakovaném použití frameworku. Odtud tedy vzniklo spojení frozen spots, neboli „zmražené místo“. Naproti tomu hot spots jsou komponenty, které spolu s kódem programátora vytvářejí zcela specifickou definovanou funkcionalitu pro konkrétní vyvíjený projekt. Z toho tedy plyne, že s každým novým projektem je tato část jiná, a proto tedy označení hot spots („horké místo“).

V objektově orientovaném programování je framework tvořen abstraktními a klasickými neabstraktními třídami. Hot spost je naproti tomu tvořen abstraktními třídami a vlastním kódem. Vlastní kód je tedy v hot spost přidán pomoci abstraktních metod.

Jak jsem již zmínil v úvodu tohoto rozboru, frameworky se ve velké míře rozšiřují a nasazují. Jako příklady z reálného světa frameworku bych rád uvedl například framework JUnit[4], který je určen k testování jednotek pro programovací jazyk Java. Další představitel frameworků je například JQuery[5], jenž je JavaScript framework s otevřeným zdrojovým kódem. Nebo také framework Spring, který spadá do kategorie aplikačních frameworků. Aplikační framework Spring je určen pro programovací jazyk Java a má otevřený zdrojový kód. Představitelů je mnoho a tudíž i jejich vypisování by bylo zbytečné, a tak bych se rád zaměřil na specifickou kategorii a tou jsou aplikační frameworky.

3.1 Aplikační Framework

Aplikační framework, někdy také označován jako webový aplikační framework, může v oblasti IT technologii a programování označovat danou softwarovou strukturu, jenž obsahuje sadu předpřipravených komponent, předloh, knihoven, postupů pro vývoj, návrhových vzorů apod. Smyslem je poskytnout stabilní a již hotový základ pro efektivní vývoj softwaru. Jednou z klíčových vlastnosti aplikačního frameworku je řešení standardních a vždy potřebných částí informačních systémů. To umožňuje zaměřit se při vývoji aplikace na specifické problémy té dané aplikace a již neřešit jednou vyřešené opakující se problémy. Aplikační framework určuje návrháři jasná pravidla co se týče struktury. Ta je totiž již předem definovaná. Spadá tedy do kategorie frozen spots. Pro vývojáře představuje aplikační framework rychlou implementaci konkrétních funkčních požadavků, než jakou nabízí obecná vývojová prostředí. Určuje a definuje jednotný způsob psaní kódu, testování či jednotné použití databáze.

Při vývoji aplikací v prostředí internetu je aplikační framework nebo spíše můžeme říci webový aplikační framework softwarovým frameworkem. Ten softwarový framework je navržen a určen pro vývoj dynamických webových stránek, webových aplikací a webových služeb[5].

[4]JUnit – jde o framework pro jednotkové testy psaný v programovacím jazyce Java[6].

[5]jQuery – je javascriptová knihovna, která klade důraz na interakci mezi JavaScriptem a HTML[7].

V dnešní době nastává taková situace, že díky rostoucím nárokům na současné webové aplikace dochází k nárůstu jejich složitosti a náročnosti. Obsahují tedy mnohém více aplikační logiky. A tak tvořit a vyvíjet tyto systémy bez výše zmíněných aplikačních struktur je velmi složité. Z toho vyplývá obrovská výhoda, jenž frameworky mají. A to je usnadnění vývoje systému a pozdější usnadnění jeho údržby. Také se snaží určit nejlepší postup vývoje webové aplikace. Všechny tyto kroky (postupy, obsahy frameworků) můžeme poté použít na jakýkoliv jiný projekt, tudíž frameworky se stavají univerzálnějšími nástroji. Největší požadavek ze strany programátor je, aby práce s frameworkem byla jednoduchá. Tím se myslí jak jeho použiti při tvorbě, tak pozdější nasazení. A další důležitým, ne-li nejdůležitějším požadavkem, je zajisté také použitá architektura. Hlavní podmínka, jenž je v tomto směru kladena, je při vývoji oddělení modelu dat od aplikační logiky a uživatelského rozhraní. Přesto, že existuje architektur návrhových vzorů několik, podmínku oddělení modelu dat od aplikační logiky a uživatelského rozhraní splňuje architektura návrhových vzorů MVC. Zatím jsme procházeli samé výhody či odlišnosti, jenž stavěly frameworky jako „nástroj“ ideální k vývoji systému. Je však také nutné podívat se na nevýhody. Jistě předem můžeme říci, že nevýhod je velmi málo, pokud tedy o nevýhody vůbec jako takové jde.

Jako jedna z mála nevýhod frameworkům bývá vytýkána doba, jenž je potřebná k prvotnímu seznámení a nastudování. Je pravdou, že může jít o delší dobu, nicméně tato doba je při další práci či použití libovolného frameworku mnohonásobně zkrácená. Dovolil bych si tuto dobu přirovnat k jednomu příkladu ze života. Pokud člověk studuje svůj první programovací jazyk, doba na pochopení je dlouhá, nicméně pokud již nějaký programovací jazyk umí, je dost věcí co jednotlivé programovací jazyky mají společné, například proměnné apod. tudíž doba na nastudování druhého jazyku je značně zkrácena znalostí již známých pojmů s předešlé doby.

3.1.1 Výhody a přednosti aplikačního frameworku:

- *Využitelnost a efektivita*
    - Standardní přístup k vývoji softwaru.
    - Předpřipravené komponenty, předlohy, knihovny, postupy pro vývoj a jiné předpřipravené nástroje. Z toho všeho vyplývá především efektivní využití lidských zdrojů, tím se myslí využití programátoru k dané věci či problematice.

- *Kompaktnost*
  - Užití jednotného stylu uvnitř aplikace. To znamená zabezpečení jednotného stylu vývoje.
  - Použitá architektura.
- *Využitelnost*
  - Navržení komponent k daným účelům aplikace.
- *Flexibilita*
  - Komponenty aplikačního frameworku využívají specifických vlastností.
- *Rozšiřitelnost*
  - Možnost rozšíření a přidaní nových komponent do aplikačního frameworku.
- *Otevřenost*
  - Možnost ladění komponent. A tudíž přechod na vyšší a novější verzi.
- *Znovupoužitelnost*
  - Není nutné vymýšlet opakovaně složité algoritmy. Použijí se již jednou vymyšlené komponenty.
- *Zrychlení a zkvalitnění vývoje aplikace*
  - A to díky tomu, že vývojový tým se soustředí pouze na vývoj části, která pokrývá specifickou obchodní logiku budoucí aplikace.

3.2 MVC Framework

Architektura MVC nabírá v posledních letech na atraktivnosti. Je to dáno také tím, že pokud chceme mít kvalitní aplikaci, musí být tato aplikace postavená na kvalitní architektuře jakou MVC je. Díky tomu firma Zend investuje nemalé prostředky nejen do vývoje PHP samotného, ale také do vývoje Zend frameworku, jenž je postaven na této architektuře. A existuje i mnoho dalších nezávislých tvůrců MVC Frameworku pro různé technologie[8].

Dovoluji si říct, že asi nejdůležitější vlastností PHP frameworků je implementace MVC architektury. MVC je zkratka třech anglických slov, ze kterých tato architektura

vychází. Jedná se o slova Model-View-Controller. Toto spojení představuje jednotlivé logické části MVC architektury. Tudíž z toho plyne, že architektura MVC je rozdělana na tři logické části nebo také komponenty. Ty lze upravovat samostatně s tím, že dopad na zbývající dvě části není téměř žádný. Mělo by tedy jít o na sobě nezávislé komponenty. Bohužel úplně tomu tak není a dojde-li k výraznější změně jedné z nich, promítne se změna alespoň částečné i v komponentě druhé .

Základní myšlenkou této architektury je oddělení logiky aplikace, datového modelu a prezentace dat v aplikaci. To vše se právě děje s pomocí tří komponent Model, View, Controller.

3.2.1 Model

Jedná se o datovou strukturu aplikace. To znamená, že jde o vrstvu, která zastupuje data (datovou vrstvu). Tato data si můžeme představit jako informace s nimiž aplikace pracuje. Ve většině frameworků pro jazyk PHP model reprezentuje jednotlivé tabulky databáze. Z toho nám tedy vyplývá, že každá jednotlivá tabulka databáze by měla mít svůj vlastní model, který jí zastupuje. V tomto modelu jsou pak umístěny činnosti, jenž bezprostředně souvisí s tabulkou. Jde o činnosti související s vložením, editací a mazáním. Z tohoto popisu tedy výchází, že model jsou data plus business logika aplikace. Může se stát, že model bude pouze sadou datových objektů bez business logiky. To je ovšem netypické.

3.2.2 View

View, v českém překladu pohled. Má za úkol poskytnout rozhraní pro prezentovaní dat uživateli. To znamená, že jde vrstvu, která prezentuje data například ve formě grafu (prezenční vrstva). Pohled umožňuje uživateli také zadávání požadavků, díky kterým vytvoří výstup a šablonu nad daty. Data, jenž potřebuje k vytvoření výstupu a šablony nad daty jsou View předány od vrstvy Model. View nemusí reprezentovat nejem graf, ale také PDF soubor, textový soubor nebo například ve webových aplikacích se převážně jedná o HTML kód s vepsaným PHP kódem, popřípadě XML[6]. Z tohoto popisu výchází závěr, že View (pohled) převádí data reprezentovaná Modelem do podoby vhodné k interaktivní prezentaci uživateli.

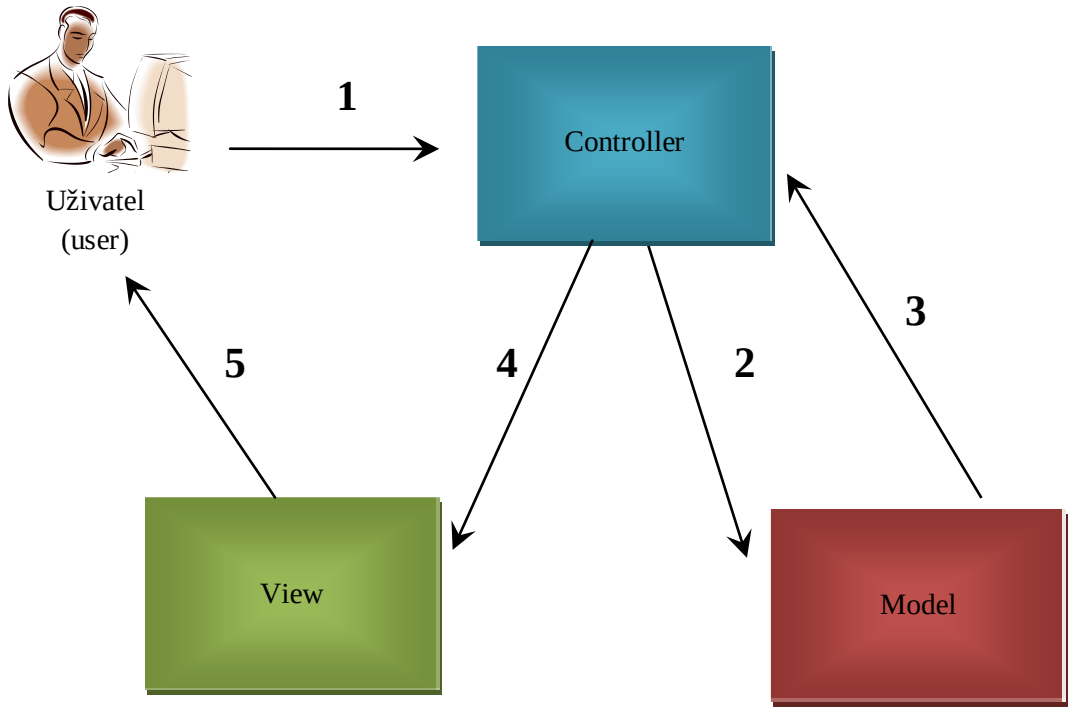
[6]XML – Extensible Markup Language. Rozšiřitelný značkovací jazyk – jde o značkovací jazyk.

3.2.3 Controller

Controller neboli v překladu ovladač či řadič, je nejhůře představitelnou částí architektury MVC. Jde ovšem o část stěžejní a nejdůležitější. Controller má za úkol řídit komunikaci mezi zbylýma dvěma komponentama. Obsahuje logiku aplikace, řídí její provoz a obsahuje požadavky od uživatele. Z toho vyplývá, že se jedná o vrstvu určenou k řízení logiky aplikace. Contoller by měl jen předávat požadavky a akce Modelu. Je tedy jakousi ústřední jednotkou, která má za úkol postarat se o celkové provázání funkčnosti aplikace. Touto komponentou se od sebe jednotlivé vzory liší nejvíce. Právě díky její návaznosti na funkčnost aplikace.

3.2.4 Funkce MVC

Poté co známe jednotlivé komponenty a jejich funkci bylo by vhodné povysvětlit, jakým způsobem probíhá komunikace mezi nimi. Tuto komunikaci budu simulovat na jednoduchém požadavku. Pro jednoduchost představení fungování komunikace komponent jsem si dovolil tuto situaci zakreslit také do obrázku[9].



Zdroj: *Vlastní*

Obrázek 1: Princip komunikace mezi komponentami

1) První krok nastane když na Controller dojde požadavek od uživatele. Zároveň s tímto požadavkem by na Controller měla dojít uživatelská data od uživatele. Tento první krok je vyvolán uživatelem, jenž provedl akci v uživatelském rozhraní.

2) Když Controller obdržel požadavek je jeho úkolem tento požadavek zpracovat a zajistit poslání žádosti Modelu pro přistup k datům.

3) Poté, co Model obdržel žádost, je na něm, aby na tento požadavek odpověděl. Tudíž Model v tomto kroku posílá zpět Controlleru potřebná data nebo odpověď o uložení informace.

4) Nyní Controller pošle výstupní data na View. Tady vidíme, že data jsou získávána z komponenty Model ovšem tato komponenta k zaslání dat nepotřebuje žádné informace o komponentě View.

5) Na závěr komponenta View vypíše data v požadovaném formátu. Tedy zobrazí se uživateli.

Na této jednoduché komunikaci je patrná nezávislost jednotlivých komponent. Výhoda spočívá v tom, že vývoj aplikace je zastřešen pod jednu architekturu. Díky této výhodě můžeme potřebné změny provádět rychleji, vývoj v týmu je přesně daný, a to z důvodu, že každý by měl vědět kde a co má hledat a poté s tím pracovat. Z toho vyplývá, že tato architekura sama vybízí k nasazení frameworku při práci v týmu.

Na závěr této architektury bych rád uvedl ve zkratce její historii. Architektura MVC vznikla v roce 1979, kdy jí popsal Trygve Reenskaug. Její první použití se vyskytlo v programovacím jazyku Smalltalk. Touto implementací byla později inspirováno celá řada projektů. Na úplný závěr k MVC architektuře bych rád připomenul, že je vhodná a také hojně používána jako architektura webových aplikací. Právě díky MVC architektuře je možné u složitých aplikací zajistit flexibilitu a spolehlivost. To je vhodné dochází-li u těchto aplikací k častějším změnám. Tento vzor však bývá často pochopen nesprávně a aplikace poté nesplňuje základní požadavek, kterým je oddělení aplikační a prezentační logiky. Možná je to i díky tomu, že základní principy člověk pochopí během pár minut, ale plně proniknout do všech detailů může trvat měsíce .

3.3 PHP Frameworky

Jak jsem již v počátku této práce zmínil, webové aplikace jsou dnes s oblibou psány v jazyce PHP. Čímž se stávají dosti nepřehledné a problém nastává zejména při údržbě nebo při nutnosti aplikaci rozšířit. Přitom existuje možnost, jak tyto problémy eliminovat a zároveň ještě získat jistou úsporu času. A jak je dnes známo s úsporou času nutného k vývoji dochází ke snížení nákladů, a to už je jistě důvodk vylepšení stávajícího PHP jazyka.

K vylepšení dojde pokud se při vývoji aplikace použijí PHP frameworky. Ty mají vlastnosti stejné jako frameworky obecně, které byly popsány v začátcích práce.

- *Značné urychlení vývoje*
- *Znovupoužitelnost kódu*
  – univerzální kód vhodný pro více aplikací
- *Zjednodušení implementace aplikace*
  – méně kódu, jednodušší řešení důležitých celků
- *Rozšíření vlastností a funkcí aplikace*
  – použití knihoven
- *Snadnější údržba či případné rozšíření aplikace*
- *Možnost práce v týmu*
  – kód je velmi přehledný a umožní práci více lidí a orientaci v ní
- *Možnost provádění debugu aplikace*
  – využití nástrojů k ladění aplikace
- *URL*
  – možnost volby hezké URL
- *Zabezpečení*
  – možnost zabezpečení aplikace proti napadení
- *Architektura projektu*
  – sjednocení architektury celého projektu
- *Open – source licence*
  – volná licence umožňující využití i v komerčních projektech

Nabídka PHP frameworků je dnes velmi pestrá. Je pouze na vývojáři samotném jakým směrem se bude ubírat jeho výběr. Díky velkému množství není však mnohdy jednoduché vybrat ten správný framework. Je mnoho aspektu, které mají vliv na pozdější použití. Frameworky se liší vlastnostmi, funkcionalitou, doplňky, ale také velikostí, kdy je nesprávné zvolit malý framework na velký projekt a naopak. Důležitou roli při výběru hraje také dokumentace nebo to, kdo se na vývoji podílí. Mám na mysli jestli je framework vyvíjen firmou a nebo komunitou. Dosti nešťastnou volbou také může být zvolení frameworku, jehož vývoj je v budoucnu nejistý. V následujících kapitolách jsem se tudíž rozhodl seznámit Vás s výběrem několika hlavními představiteli.

3.3.1 Důležité vlastnosti PHP frameworků

Tím, čím jsou jednotlivé frameworky odlišné a zajimavé jsou jejich vlastnosti a schopnosti. Proto je důležité si důkladně nejprve prostudovat dané možnosti a teprve poté zahájit vývoj aplikace. Pro lepší orientaci je zde vybráno několik důležitých bodů, na které je vhodné se při výběru zaměřit.

- Návrhový vzor
  - Většina dnešních PHP frameworků podporuje MVC architekturu, popřípadě její obdobu jako MVP. Zde začíná základ vhodné volby.
- Verze PHP
  - Je dobré myslet také na to, jakou verzi framework podporuje. Aktuální verzi PHP 5 (PHP 5.3.2) podporuje opět většina frameworků. Opak však nastává při potřebě použití PHP 4. Nicméně tato volba je již dnes vcelku nevyhovující.
- Podpora databází
  - Podpora různých databází.
- Šablony
  - Podpora šablonovacího systému, který pomáhá oddělit HTML od aplikačního kódu.

- Ajax[7]
    - Podpora dnes populárního jazyka Ajax.
- Zabezpečení
    - Druhy podpory zabezpečení. Ty umožňují odražení útoku hackera a tím mu zabrání proniknou do aplikace a získat tak citlivá data.
- Volba komponent
    - Díky podpoře komponentového přístupu je možné framework obohatit a různé funkcionality.
- Caching
    - Možnost ukládání dat co cache (dočasné paměti).
- Podpora doplňků (modulů)
    - Podpora možnosti rozšíření frameworku pomocí doplňku o emaily a podobně

[7]AJAX – Asynchronous JavaScript and XML. Asynchronní JavaScript a XML – jde o technologii vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovu načítání[10].

4 Přehled PHP frameworků

Tato kapitola se zaměří na přehled představitelů z řad frameworků. Těch je velké množství a tak jsem zvolil pouze ty nejznámější. Výběr jsem se snažil provést tak, aby vlastnosti a funkce frameworku pokryly standardní požadavky moderní aplikace. I přesto, že frameworků existuje celá řada, pokud se na výběr zaměříme detailněji, zjistíme, že našim požadavkům vyhovuje pouze jistá část. Důležitou složkou každého z nich je dokumentace. Pokud je dokumentace na špatné úrovni, práce s frameworkem trvá mnohem déle, protože je třeba dlouze vyhledávat parametry funkcí či řešení jiných úskalí. Tudíž na tuto část jsem se v tom přehledu zaměřil o něco detailněji. Kromě dokumentace se zde také můžete seznámit s důležitými a jedinečnými vlastnosti každého z frameworku.

4.1 CakePHP

CakePHP je založen na jednoduchosti. Vychází a inspiroval se frameworkem Ruby on Rails[8], jenž je frameworkem pro programovací jazyk Ruby. CakePHP je vysoce produktivní open source webový aplikační framework[11]. Při jeho nasazení dojde k usnadnění vývoje aplikací v jazyce PHP. Tento framework je určen pro vývoj webových aplikací v jazyce PHP a také je v jazyce PHP napsán. Celý je postaven na vzoru MVC a nutí vývojáře ten vzor používat. Díky tomu je kód přehledný a snadno pochopitelný i pro jiného vývojáře.

Vznikl v roce 2005 pod rukama autora Michala Tatarynowicz. Nyní je jeho vývoj zastřešen firmou Cake Software Foundation.

Aktuální verze CakePHP nese označení CakePHP 1.2.5.. CakePHP si klade za cíl pracovat strukturovaně, rychle, bez ztráty flexibility a odstranit jednotvárnost z vývoje webových aplikací. Má dobře navrhnuté základní konvenční chování, i přesto však je u něj zachována výrazná variabilita. Zaměřuje se především na přehledný a krátký zdrojový kód, kvalitní databázovou vrstvu typu active record, promyšlené view helpery a pokrývá běžné operace, které programátor potřebuje.

CakePHP je vhodný framework například pro vývoj e-shopu nebo standardního CMS.

[8]Ruby on Rails – je plnohodnotný framework pro vývoj webových aplikací napojených na databázi. Využívá MVC architekturu a vychází z programovacího jazyka Ruby.

Dokumentace:

Tento framework má za sebou silnou a početnou komunitu, proto i zdrojů, ze kterých je možno čerpat, je dostatek. Na stránkách frameworku (http://cakephp.org/) je detailně zpracován popis API[9]. Také zde nalezneme kvalitní dokumentaci. Dokumentace u toho frameworku je na výtečné úrovni. Je nazývána CookBook neboli v překladu znamenající „kuchařka“ [12]. Nejde o žádnou strohou příručku. Vše je v ní zaznamenáno k dané problematice bez zbytečného textu odvádějícího pozornost od tématu. Příručka je psaná příjemnou formou a je doplněna spoustou názorných příkladů, díky kterým dojde snáz k pochopení. Dokumentace je koncipována stylem, který provází uživatele při práci s tímto frameworkem od úplných začátků krok po kroku.

Seznam vlastností CakePHP:

- Flexibilní licence.
- Kompatibilita s PHP ve verzi 4 i 5.
- Integrovaný CRUD sloužící pro práci s databází, zjednodušené dotazy
- MVC architektura.
- Flexibilní podpora ACL.
- Flexibilní Caching Pohledu.
- Zabudovaná validace dat.
- Fungování na libovolném hostingu s minimální, nebo žádnou konfigurací Apache.
- Jazyková lokalizace aplikace.
- Request dispatcher s čistými URL a cestami, tedy mnody dobře vypadající uživatelské URL.
- Rychlé a flexibilní šablony (syntaxe PHP, s helpery).
- Automatické generování kódu (Administrace systému).
- Helpery zobrazovací vrstvy neboli podpora Pohledu pro AJAX, JavaScript, HTML formuláře a další.
- Použití metody scaffolding[10].
- Email, Cookie, Session, bezpečnost, komponenty pro obsluhu požadavku.

[9]API - Application programming interface – jde rozhraní o programování aplikací. Obsahuje procedury, funkce a knihovny, které může programátor využít při programování.

[10]Scaffolding je programovací metodou stavějící na databázových aktivitách dané aplikace.

4.2 Prado

Prado je založen na komponentovém přístupu a programování událostí. Což je hlavní vlastností tohoto webového aplikačního frameworku[13]. To však také znamená, že nejde o úplně klasický PHP framework. Události nám u tohoto frameworku umožní propojit šablonu stránky s PHP kódem. Jako ostatní PHP frameworky, tak i tento je postaven pro podporu jazyka PHP. Podporuje verzi PHP 5.1.0 nebo vyšší. Prado je stejně jako CakePHP open source webový aplikační framework. Inspirace a základní myšlenky tohoto frameworku vycházejí z Borland Delphi a z ASP.NET frameworku.

PRADO je zkratka pro Rapid Application Development Object–oriented neboli ve volném překladu znamenající rapidní vývoj objektově orientovaných aplikací v PHP. Za vznikem tohoto frameworku stojí Qiang Xue. Ten začal framework vyvíjet v roce 2004 a díky tomu prošel již dlouhým vývojem. Současnou verzí je verze nesoucí označení 3.1.6 a byla vydána 22. července 2009. Tato verze se hodí pro vývoj větších webových aplikací. Pro tento vývoj je určeno používání hotových komponent, kterým lze změnou parametru přizpůsobovat jejich chování, reakce na události naprogramováním obslužných funkcí a začleněním do výsledných stránek aplikace.

Dokumentace:

Dokumentaci nalezneme na oficiálních stránkách tohoto frameworku (http://www.pradosoft.com/).

Je psaná stručně a po různých sekcích, což může připomínat například styl stránek www.wikipedia.org.

Ne každému tak může vyhovovat. Jeden z důvodu je také horší orientace a zdlouhavé vyhledávání dané věci potřebné k nalezení. Naopak výhodou je stažení manuálu neboť je na stránkách umístěn ve formě kniha s formátem .pdf. Dále také na stránkách frameworku nalezneme dokumentaci API a popis tříd. Obrovským přínosem je však velké množství dobře zpracovaných tutoriálů, jenž zde nalezneme ve formě HTML prezentace a videa. V těchto tutoriálech můžeme najít tvorbu jednotlivých funkcionalit, ale také tvorbu internetového systému jakým je blog. Právě tyto tutoriály budou uživateli slouží více k nastudování tohoto frameworku než samotná dokumentace.

Seznam vlastností Prado:

- MVC architektura.
- Objektově orientovaný.
- Znovupoužitelnost díky pravidlům psaní zdrojového kódu, která jsou stanovená pro tento framework.
- Událostmi řízené programování, různé aktivity jsou zachyceny jako serverové události.
- Podpora AJAXu.
- Silná podpora a kompatibilita s databází, MySQL 4.1+, PostgreSQL 7.3+, SQLite 2 a 3, MSSQL 2000+, Oracle (alpha) [14].
- Podpora internacionalizace a lokalizace díky I18N a L10N.
- Přidání kontroly CAPTCHA k formuláři.
- Použitelné cachování, to znamená statické ukládání komponent, které byly inicializovány.
- Přizpůsobitelná obsluha chyb a výjimek.
- Hlášení chyb a výjimek a přizpůsobení chybových hlášení.
- Rozšiřitelná autentizace a autorizace.
- Zabezpečení webových aplikací, jako například prevence cross-site(XSS), ochrana cookies.
- Adaptace již existující práce, díky tomu je možné zavádění nejnovějších knihoven a tříd vývojářem.
- HTML komponenty.
- Podpora webových služeb (JSON[11] a SOAP[12]).
- Vysoce použitelné webové komponenty, jako validace vstupu, tvorba průvodce (wizard) atd.
- Podpora dokumentace.

[11]JSON - JavaScript object notation – jde se o odlehčený formát pro výměnu dat. Je to textový formát zcela nezávislý na programovacím jazyce, využívající konvence programovacího jazyka C[15].

[12]SOAP – Simple Object Access Protocol. Jednoduchý objektový přístupový protokol – jde o protokol pro výměnu zpráv založených na XML přes síť pomocí HTTP[16].

4.3 Nette Framework

Nette framework je výkonný webový aplikační framework sloužící pro tvorbu webových aplikací a služeb. Je napsán v PHP 5 a plně využívá objektově orientovaného programování. Jde o open – source knihovnu. Podporuje a využívá vlastností PHP 5.3, událostmi řízené programování a použití komponent. Jeho jednotlivé časti jsou na sobě nezávislé, takže je jde používat i samostatně. Nette framework se zaměřuje na snížení bezpečnostních rizik.

Jde o ryze český framework za jehož vývojem stojí David Grudl. Vznikl v roce 2004 avšak teprve v roce 2008 byl uvolněn jako open source a tedy i zpřístupněn veřejnosti. Jeho licence, která vychází z BSD[13], patří k těm nejvolnějším[17]. Má velmi početnou komunitu především v České republice. Možná je to také dáno tím, že je nejen velmi kvalitní a rychlý, o čemž svědčí i různé porovnání, jenž lze v síti internetu najít, ale také to, že je kompletně v češtině. Tento framework využívá řada tuzemských společností a tudíž je vhodný jak na nasazení menších projektů, tak i těch středních, jako například blogy a eshopy. Nyní se o rozvoj Nette framework stará organizace nesoucí název Nette Foundation.

Dokumentace:

Dokumentaci nalezneme na oficiálních stránkách tohoto frameworku (http://nettephp.com/cs/). Je přehledně roztřízená, takže se v ní přehledně hledá. Na stránkách také nalezneme dokumentaci API. Obrovskou výhodou u této dokumentace je, že je v češtině. To může pomoci tuzemským vývojářům i přesto, že anglický jazyk je dnes brán za samozřejmost. V dokumentaci je možné také nalézt průvodce frameworkem a seznámení s jeho funkcemi a doplňky. Jistou nevýhodou je to, že jak stránka, tak i dokumentace se stále nachází ve stadiu vývoje a tak ne vše je funkční. Bohužel to je dáno tím, že ještě stále se jedná o nový framework. Dost zdrojů se však nachází i mimo oficiální stránky a tak někdy je lepší při studiu frameworku navštívit i jiné zdroje.

[13]BSD Licence – jde o volnou softwarovou licenci, která patří mezi nejsvobodnější. Umožňuje volné použití či síření a to i v komerční sféře. Pouze je nutné uvést autora a informace o licenci.

Seznam vlastností Nette frameworku:

- MVC architektura.
- Objektově orientovaný.
- Znovupoužitelnost díky pravidlům psaní zdrojového kódu, která jsou stanovená pro tento framework.
- Událostmi řízené programování .
- Podpora AJAXu.
- Použitelné cachování.
- Podpora dokumentace.
- Velmi výkonný.
- Podpora pro práci s databází, ta je realizovaná pomoci knihovny Dii, což je databázová vrstva určená pro PHP 5.
- Pohledy používají vlastní šablonovací systém.
- Základní validační pravidla.
- Helper pro XHTML prvky.
- Snadná práce s URL, tím mám na mysli obousměrný přepis a tedy, že se odkaz na akci, která se má provést. Tím se URL vytvoří samo. URL struktura je pak nezávislá na zbytku aplikace.
- Snížení bezpečnostních rizik.

4.4 Symfony

Symfony je webový aplikační framework psaný v jazyce PHP. Je určen pro vývoj webových aplikací založených na PHP 5. Jako předchozí frameworky i tento vychází z návrhového vzoru MVC. Framework obsahuje celou řadu nástrojů a knihoven, jenž usnadňují a automatizují vývoj a údržbu webových aplikací a také činnosti spojené s opakovaným použitím stejných částí kódu. Symfony Framework je určen především tvorbou rozsáhlých aplikací. Díky tomu je často využíván firmami pro vývoj robustních celopodnikových systémů. Framework vychází z největší úspory psaní kódu programátorem. Velká část kódu je pak rozdělena do konfiguračních souborů. Ty jsou psány ve velmi úsporném a čitelném syntaktickém zápise. Tento zápis se nazývá YAML. Symfony se svými minimální nároky je vhodný Framework na jakékoliv konfiguraci. Je snadno snadno instalovatelný a lze jej nainstalovat na všechny hlavní operační systémy. Stačí tedy mít Unix

nebo Windows s webovým serverem a podporou PHP 5. Je také kompatibilní s většinou dostupných databází.

Framework má v sobě zahrnuto spoustu pluginů, které mají programátorovi co nejvíce ulehčovat práci při standardních postupech programování na webu. Programátor má také kontrolu nad konfigurací, od adresárové struktury až po cizí knihovny. Může si tedy téměř vše přizpůsobit. Ovšem nevýhodou takto obsáhlého frameworku je to, že ovládnout ho trvá zřejmě ze všech velkých frameworků nejdéle.

Symfony vznikl v roce 2005. Vyvinut byl společností Sensio Labs, jenž také nadále podporuje jeho vývoj. Aktuální verze nese označení Symfony 1.4 a vyšla v listopadu 2009. Je postavená na PHP ve verzi 5.2.4.

Dokumentace:

Dokumentaci nalezneme na oficiálních stránkách tohoto frameworku (http://www.symfony-project.org/**)**. Dokumentace u tohoto frameworku se řadí k těm lepším s dobrou podporou. Na stránkách tohoto frameworku nalezneme jednak API dokumentaci, dále pak kvalitní dokumentaci a v neposlední řadě je zde spousta tutoriálu, jenž jsou nachystány na postupné nastudování. Toto nastudovaní má strukturu jednu hodinu denně pod dobu 24 dnů.V tutoriálech lze najít například návod na vytvoření prvního projektu, návod na použití formulářů, návod na použití AJAXu a různé další návody. Jak dokumentace tak tutoriály jsou popsány v pěti světových jazycích.

Seznam vlastností Symfony:

- MVC architektura.
- Oddělení objektového modelu a MVC.
- Cachování.
- Podpora AJAX.
- Podpora internacionalizace (I18N).
- Generování „hezkých“ URL (strojově čitelná URL, jenž lze využit například pro SEO).
- Validace formulářů.
- Jednoduché šablonování.
- Debugování.

- Možnost vlastního doplnění Symfony.
- Složitější konfigurace a práce s frameworkem.
- Použití Plutonu.
- Minimální nároky na konfiguraci.
- Kompatibilní s většinou databází.
- Možnost použití na všech hlavních operačních systémech.
- Libovolný počet nastavitelných kontrollerů (například development-controller) [18].
- Možnost správy aplikace z příkazové řádky.
- Spolupráce s Propelem – jednou z prvních ORM knihoven v PHP (jelikož už byla ukončena došlo odstranění závislosti na Propelu do pluginu).

4.5 CodeIgniter

CodeIgniter je webový aplikační framework psaný v jazyce PHP. Je určen pro vývoj webových aplikací založených na PHP 4.1. I tento frameworky vychází z návrhového vzoru MVC. I přesto, že se může zdát, že jde o zastaralý framework, díky verzi PHP z které vychází není tomu tak. Běží i na PHP 5, bohůžel však nevyužívá žádné její nové vlastnosti. To však vývojářům nahrává, neboť můžou do tohoto frameworku psát vlastní kód s vlastní funkcionalitou, který bude těžit z vlastností PHP 5. Tím by pak vznikla jakási nástavba frameworku. Bohůžel autoři prozatím nemají v plánu vypustit verzi, jenž bude vycházet z PHP 5, a to díky malé rozšířenosti podpory této verze na komerčních hostingových službách. CodeIgniter umožňuje psát aplikace velice rychle a efektivně a téměř bez nutnosti jakéhokoliv nastavování. Podpora jazyka PHP je od verze 4.3.2 a výš. K provozování a bezproblémové funkčnosti aplikace je potřeba aplikaci provozovat na hostovaném webovém serveru, jenž je schopný zpracovávat PHP skripty, jako je Apache nebo LiteSpeed. Jedná se menší framework co do rozsáhlosti. Tím se stává velmi jednoduchým na používání a práce při vytváření aplikací je snadnější. I přes svou velikost je tento framework vhodný k vývoji středních i větších aplikací. Právě pro jednoduchost CodeIgniteru je chod stránek v něm vytvořených mnohem rychlejší, než u ostatních frameworků. Samotný framework je vyvíjen jako odlehčený, to znamená, že jádro systému závisí jen na několika malých knihovnách. Další knihovny si vývojář může doplňovat dle potřeby za chodu frameworku.

Za celým vývojem CodeIgniter frameworku stojí americká společnost EllisLab. K vývoji však také značně přispívá silná komunita, jenž se nachází okolo tohoto frameworku. První verze frameworku byla nasazena v únoru roku 2006. Aktuální verzi je verze nesoucí označení 1.7.2[19]. Tato verze obsahuje 26 různých knihoven ve formě

samostatných tříd, které svou nabízenou funkcionalitou pokrývají standardní a často řešené úlohy. Licence frameworku vychází z BSD licence.

Dokumentace:

Dokumentaci nalezneme na oficiálních stránkách tohoto frameworku (http://codeigniter.com/**)**. Dokumentace je velmi podrobná a řadí se k těm nejlépe zpracovaným. Je rozdělená na několik sekcí a díky tomu je velmi přehledná a nalezneme v ní spoustu užitečných rad a triků. Vždy se odkazuje na poslední aktuální verzi tohoto Frameworku. Na stránkách také nalezneme řadu tutoriálů, jenž jsou zpracovány ve formě videa. Tyto tutoriály umožní a ulehčí práci při prvním seznámení s tímto frameworkem, neboť jsou postaveny na ukázkách, jenž provedou uživatele prvotním základním nastavením a použitím. Také je možné v tutoriálech nalézt ukázkové aplikace, jako například „tvorbu blogu za 20 minut“, kde je názorně ukázaná práce s databází. Díky velmi silné komunitě, jenž se kolem tohoto frameworku nachází, bych doporučil také navštívit na těchto stránkách fórum, kde je plno rad a triků, jenž je možné při vývoji aplikace využít. Bohužel na stránkách chybí API dokumentace, což může trochu vadit, ovšem nejedná se o nic tak zásadního. Je totiž možné navštívit jiné zdroje v síti internetu a tam najít vše potřebné. Tím chci naznačit, že oficiální stránka není jediný zdroj s řešenými příklady a informacemi a uživatel má tedy možnost navštívit i jiné zdroje. Ovšem povětšinou v anglickém jazyce. Díky takto obsáhlým materiálům se tento framework snadno učí. Na závěr bych ještě rád poukázal na to, že na oficiálních stránkách je možné mimo dokumentace a tutoriálů také nálezt dokument, který popisuje pravidla, jak vhodně a správně psát aplikaci postavenou nad tímto frameworkem.

Seznam vlastností CodeIgniter:

- MVC architektura.
- Silná podpora a kompatibilita s databází, MySQL (4.1+), MSSQL, PostgreSQL, Oracle, SQLite, ODBC. Pro práci s databází používá upravenou třídu Active Record[14]. Tato třída však neumí pracovat s relacemi.

[14]Active Record – je návrhový vzor používaný pro práci s datovými zdroji v relační databázi.

- Existence Plutonu.
- Práce s Excel, generátory PDF či jednoduchými obrázky .
- Obsahuje standardní helpery.
- Generování „hezkých“ URL (ale pouze pomocí vyžadovaného mod_rewrite).
- Validace dat formulářů.
- Callback - díky němu lze snadno do validace zapojit i vlastní funkce.
- Možnost implementace šablonovacího systému Smarty.
- Podpora jazykové lokalizace.
- Cachování.

4.6 Zend Framework

Zend Framework je objektově orientovaný, webový aplikační framework. Stejně jako ostatní výše zmíněné frameworky i tento je implementován v jazyce PHP, a to ve verzi 5. Také jako ostatní popsané frameworky i tento je open source. Za vývojem tohoto Frameworku stojí společnost Zend Technologies Ltd. Pro připomenutí podotýkám, že Zend Technologies Ltd je společnost, která stojí za jádrem PHP 4 a jeho následných verzí. Zend Framework je licencovaný pod New BSD licence a je vyvíjen s ohledem na jednoduchý vývoj webových aplikací[20]. Jeho síla spočívá ve vysoce modulární MVC architektuře, díky níž je kód více a opakovatelně použitelný. Užitím této architektury, umožňuje vývojářům použití jen těch komponent, které jsou nezbytně nutné k řešení dané problematiky při vývoji daného systému. Ovšem částečná závislost mezi komponentami existuje. Tento framework v sobě zahrnuje celou řadu komponent ať už komponenty pro MVC aplikace, autorizaci autentifikaci, tak implementuje i různé druhy cache, validátorů pro uživatelská data, jazykové komponenty a mnoho dalších.

Za celým vývojem Zend Frameworku, jak jsem již zmínil stojí Zend Technologies Ltd. K vývoji však také značně přispívá silná komunita, jenž se nachází okolo tohoto frameworku. Ta je takto rozsáhlá díky kvalitě, jednoduchosti a také tím, že se jedná o jeden z prvních frameworku pro jazyk PHP. První verze frameworku začala být vyvíjena na počátku roku 2005. Jde o období, kdy různé Frameworky pro různé jazyky získávají na popularitě. První oficiální oznámení a představení tohoto Frameworku proběhlo na první Zend Conference. Za celým tímto projektem také stojí jeden hlavní šef vývojář, a tím je Will Sinclair. Aktuální verzi je verze nesoucí označení 1.9.7.

Dokumentace:

Dokumentaci nalezneme na oficiálních stránkách tohoto frameworku (http://www.zendframework.com/). Dokumentace u tohoto frameworku se řadí k těm nejlépe zpracovaným s dobrou podporou a základní příručkou v šesti světových jazycích. Na stránkách tohoto frameworku nalezneme jednak API dokumentaci, dále pak kvalitní dokumentaci a v neposlední řadě je zde spoustu video tutoriálu a prezentací, jenž jsou nachystány na postupné nastudování. Orientace v dokumentaci je snadná a je také doplněna velkými množství příkladů. Jelikož jde o jeden z nejpopulárnějších frameworku je komunita kolem něj obrovská. Jedná se o celosvětově rozšířený framework, tudíž jazyková bariéra by zde neměla být problémem. Na internetu lze díky tomu nalézt existující velká fóra, která obsahují spoustu rad, tipů, triků, příkladů a v neposlední řadě řešených problémů. Jelikož Zend framework má velké zastoupení i mezi českými vývojáři, lze tedy najít mnoho návodů a řešených problémů právě v češtině.

Seznam vlastností Zend Framework:

- MVC architektura.
- Modulární architektura – minimalizace závislosti mezi komponentami .
- Silná podpora a kompatibilita s databází, MySQL (4.1+), MSSQL, SQLite, Oracle, PostgreSQL, IBM DB2. Pro přístup do databáze využívá PDO.
- Jednoduché šablonování – Díky využiti šablonovacího systému SMARTY.
- Rozšiřitelná implementace MVC - s podporou layoutů a šablonovacím systémem.
- Cachování – Flexibilní cache sub-systémy s podporou mnoha typů backendů, jako paměť nebo soubor.
- Všechny komponenty jsou plně objektově orientované a vyhovují direktivě E_STRICT.
- Helpery – obsah standartních Hesperů.
- Pohledy – můžou být klasické soubory tvořeny kombinací HTML a PHP mající příponu phtml[21].
- Součást modulů.
- Validace – obsahuje i nadstandardní validační metody, jako například ověření, zda-li jde o IP adresu, hexadecimální číslo nebo datum.
- Podpora AJAX.

5 Podrobnější rozbor jednotlivých frameworků

Jestliže předchozí kapitola měla pouze nastínit přehled možných frameworků a jejich charakteristiku, tak tato kapitola má za úkol projít podrobněji dvěma frameworky. Volba pro podrobnější popis byla volná a tak došlo k vybrání Prado Frameworku a Nette Frameworku. Možná nastává otázka, proč právě tito dva zástupci. Důvodů bylo hned několik. Jelikož jedna z věcí, která mě na frameworcích zaujala, je komponentový přístup. Díky tomuto přístupu je nám umožněno tvořit software interakci již hotových části. Což je defakto opak k objektově orientovanému programování, které chce, aby objekty odpovídaly tomu, co software skutečně obsahuje.

Možná tedy proto mě zaujal Prado framework více než jiní představitelé. Prado je založen právě na komponentovém přístupu a programování události. To umožňuje při vývoji využívat hotové komponenty, které po přizpůsobení parametrů určují přesné chování, jenž je vyžadováno od aplikace.

Druhým zástupcem je pak Nette Framework. Tento framework jsem jsi vybral opět kvůli tomu, že je založen na použití komponent a využití událostmi řízeného programování. Ovšem to nebyl důvod jediný. Tento framework jsem zvolil také právě proto, že jde o ryze český Framework. A je tedy vhodnou volbou právě pro tuzemské vývojáře. Myslím si, že jeho vývoj postoupil za posledních pár měsíců značně kupředu a tím se dostal do popředí PHP frameworků.

5.1 PRADO

5.1.1 Výhody, přednosti a vlastnosti

Znovupoužitelnost - Tento framework je díky intuitivnímu kódu velmi vhodný pro použití při práci v týmech. Psaní kódu je ohraničeno jistými pravidly, které je doporučeno dodržovat. Znovupoužitelnost využijeme i při opakované tvorbě odlišných aplikací. Již nebude nutné opětovně psát jednou vymyšlené funkce a části aplikace postačí pouze změna atributů.

Událostmi řízené programování - Tato vlastnost je typickým rysem toho frameworku. Díky tomu jsou různé aktivity zachyceny jako serverové události a vývojáři se mohou zaměřit především na interakci uživatele.

Silná podpora a kompatibilita s databází - Od verze 3.1 byl framework vybaven kompletní databázovou podporou, jejíž použití je intuitivní. Tato podpora databází je velmi široká. Zahrnuje všechny důležité představitele jako MySQL 4.1+, PostgreSQL 7.3+, SQLite 2 a 3, MSSQL 2000+, Oracle (alpha). V závislosti na složitosti celé aplikace je možné si zvolit, jaký bude přístup do databáze. Mužeme volit jednoduchý přístup prostřednictvím PDO (PHP Data Objekt), popřípadě známý Active Record nebo kompletní objektově mapované schéma SQLMap[22].

Podpora značkovacího jazyka XHTML - Kód generovaným tímto frameworkem je v XHTML a plně splňuje požadavky W3C konsorcia.

Podpora AJAXu - Díky vestavěné podpoře AJAXu již nebude nutné psát jednotlivé řádky kódu v JavaScriptu. Právě to nám umožní vytvořit AJAXovou aplikaci pomocí několika řádků kódu.

Zabezpečení webových aplikací - Prado využívá hned několik metod, jak zabezpečit aplikaci. Díky tomu je odolnější proti napadení a proniknutí k citlivým datům. Framewrok je odolný například proti útoku Cross-site scripting (XSS).

Cachovaní - Díky využití cachování je Prado Framework velmi výkonný a rychlý

5.1.2 Požadavky na systém

Systémové požadavky na platformu jsou standardní jako nainstalovaný webový server podporující minimálně PHP5.1 a vyšší. Ovšem chceme-li pracovat vždy s maximálním výkonem a využitím frameworku, je doporučeno mít aktuální verzi PHP. Volba webového serveru není nijak omezena. Osobně jsem při práci využíval ApacheWeb Server. Co se týče databázových serverů i s těmi není u tohoto frameworku žádný problém a standardně podporuje všechny hlavní představitele jako například SQLite a podobně. Pro vývoj je vhodné využít již zmíněný ApacheWeb Server a tehdy pak je root dostupný na obvykle adrese http://localhost. Druhou volbou může být nahrání kompletního balíčku na webový server. Poté už jen zbývá zadat do prohlížeče adresu skriptu. Následně dojde k zobrazení úvodní stránky frameworku s manuálem a ukázkami programů.

5.1.3 Instalace

Instalace Prado frameworku je velmi jednoduchá. První a základním krokem je nutnost stáhnout si aktuální distribuci na svůj lokální disk. Pokud se pozorně podíváme na stránky výrobce (http://pradosoft.com/download/) zjistíme, že ke stažení je několik verzí. Tyto verze jsou sestupně řazeny od aktuální až po nejstarší. Aby nedošlo k omylům a pozdějším problémům je možné u každé verze nalézt tak zvaný Change log. V tomto souboru jsou zaznamenány veškeré úpravy, jenž daná verze prodělala oproti té předchozí.

A nyní se přesuneme již k samotné instalaci. Poté co jsme stáhli archiv s aktuální verzí je nutné je rozbalit. Následně rozbalenou distribuci nahrajeme na webový server, popřípadě přesuneme do adresáře, který je dostupný z adresy http://localhost. Nyní jsme se dostali do závěrečné fáze, kdy nám zbývá spuštění samotné distribuce.

Pokud chceme zkontrolovat je-li framework plně připraven k využití, můžeme provést kontrolu konfigurace. K tomu nám poslouží Requirements-Checker nástroj. Ten je možné spustit na následující adrese, kterou zadáme do prohlížeče.

```
http://localhost/prado/ requirements/index.php
```

Po ověření a případném vyladění je z důvodu zabezpečení vhodné Requirements Checker ze serveru odstranit.

5.1.4 Adresářová struktura

Díky instalaci nám vznikla prvotní adresářová struktura. Tu lze u Prada frameworku upravovat a vytvořit si tak strukturu svou. Nicméně měnit tuto strukturu bych doporučil až zkušenějším uživatelům. Při změně je důležité se řídit jistými pravidly a také je nutné zachovat správné cesty do adresářů ve spouštěcích souborech.

Základní vnitřní adresářová struktura má takovouto strukturu. Ta se bude postupně rozšiřovat či měnit dle potřeb a také při nahrávání potřebných knihoven.

```
NazevProjektu/example/assets/
                    /protected/pages/
                    /runtime
                    /idnex.php
                    /prado-3.1.7.r2783
```

Mezi hlavní části adresářové struktury patří adresáře nesoucí název /assets, /protected,, /prado-3.1.7.r2783.

Prvním z adresářů je /assets. Ten je prázdný. Ovšem je nutné pamatovat na to, že je třeba mít pro tento adresář nastavena práva pro zápis. To z toho důvodu, že tento framework bude do něj ukládat dočasné soubory nutné pro svůj chod.

```
NazevProjektu/example/assets/
```

Druhý adresář nese název /protected. Jde o jeden z nejdůležitějších adresářů při vývoji webových aplikací. V tomto adresáři můžeme nalézt skripty, šablony stránek aplikace, konfigurační soubory zjednodušeně pohledy a controllery. Jde o veškeré zdrojové soubory, díky tomu je také nutné dodržovat jisté zabezpečení. Téměř nutností je adresář zabezpečit proti neoprávněným přístupům a také by tento adresář neměl být přístupný z prohlížeče. O tuto ochranu se stará soubor .htaccess, jenž je umístěn uvnitř adresáře.

```
…/…/protected/pages/Home.page      - obsahuje vizuální XHTML stránku s
                                     komponentami. Uživateli bude zobrazena při
                                     načítaní zobrazení stránky.
                  /Home.php        - obsahuje programovou část souboru Home.page.
                                     Je nutné aby byla zachována hierarchie těchto
                                     dvou souborů. A to tedy znamená, že musím být
                                     na stejné adresářové úrovni
          /runtime/.htaccess       - zajišťuje ochranu adresáře protected.
```

Posledním ze základní řady adresářů je adresář /prado-3.1.7.r2783. Ten obsahuje aktuální distribuci Prado frameworku se všemi jeho součástmi. Jak je již podle názvu patrné aktuální verzí je Prado 3.1.7.

```
NazevProjektu/example/ idnex.php    - vytváří instanci aplikace a slouží k jejímu spuštění
                  /prado-3.1.7.r2783 - aktuální verze frameworku
```

5.1.5 Struktura a práce s frameworkem

Pro snazší představu si zde popíšeme jednotlivé části návrhového vzoru MVC. Také si v tomto popisu řekneme, co je důležité v jednotlivých částech dodržovat či nastavovat.

Práce s Modelem:

Jak je nám již známo, model má za úkol umožnit práci s daty v tabulce databáze. Přičemž platí, že každá jednotlivá tabulka databáze by měla mít svůj vlastní model, který jí zastupuje. To je v tomto frameworku zajištěno vždy jednou třídou, která zastupuje jednu tabulku databáze. Tato daná třída je pak pouhým rozšířením třídy TActiveRecord, kdy pomocí dědění je zajištěna práce s daty v tabulce.

Každá třída pro danou tabulku musí mít v deklaraci uveden název tabulky a její atributy. Struktura v dané třídě musí plně odpovídat fyzické struktuře skutečné databáze.

Práce s View:

View je další ze tří částí MVC architektury. View, neboli v překladu znamenající pohled, má za úkol poskytnout rozhraní pro prezentovaní dat uživateli. Stejně tak slouží i k zadávání požadavků v podobě formulářů a následnému uložení dat. View je v Prado reprezentován sadou komponent. Tato sada je obsáhla. Práce s komponentami není nijak složitá. Jak již bylo naznačeno v kapitole adresářové struktury, o vizualizaci aplikace se starají soubory s příponou .page. K těm je nutné vždy doplnit i programovou část. O tom, ale až v další části.

Práce s Controllerem:

Controller je poslední částí MVC architektury. Jde o část, která má za úkol postarat se o celkové provázání funkčnosti aplikace. Přesněji tedy má za úkol předávat požadavky a akce Modelu. Controller u Prado frameworku snadno poznáme. Má totiž totožný název jako soubor s příponou .page, jen přípona bude nahrazená za standardní .php. Jeho název je tedy totožný z další částí MVC architektury a to View. Pokud se dobře zamyslíme zjistíme, že stejně jako pohled tak i controller nebude jediný. A to právě proto, že jej je nutné zajistit pro každý View. Výsledné třídé této komponenty se nacházejí v adresáři pages. Ty pak opět dědí z jedné hlavní třídy a tou je TPage.

5.1.6 Zajímavé vlastnosti a funkce

Stejně jako jiné frameworky, tak i Prado Framework obsahuje celou řadu zajímavých komponent. Pro usnadnění interakce mezi komponentami je implementováno paradigma pro událostmi řízené programování. Pomoci této implementace lze nastavovat chování komponent dle daných potřeb. Tato rozšíření frameworků nejsou jeho součástí, a tak při jejich použití je nejprve nutné komponenty dohledat a poté nainstalovat. Obsáhlou nabídku lze nalézt přímo na stránkách www.pradosoft.com. U tohoto frameworku lze pak komponenty snadno rozlišit od jiných doplňků. A to proto, že jejich název vždy začíná velkým písmenem T.

Jelikož doplňků je celá řada, bylo by složité a zdlouhavé Vás se všemi seznámit. Pokusíme se tedy podívat pouze na pár z nich.

Komponenta pro tvorbu zjednodušených přehledových tabulek (DataGrid):

Tato komponenta umožňuje při použití několika řádku kódu vytvořit interaktivní webové stránky. Pomocí <com:TDataGrid> je možné snadno implementovat výpis představující tabulku v databázi. Ta pak může obsahovat například funkce pro řazení, editace, vkladání a mazání dat. Samozřejmě výčet funkcí není konečný. DataGrid umožňuje kromě zjednodušené tvorby přehledových tabulek i přehlednou vizualizaci uživateli. Tato vizualizace dat je přehledná. Vhodné ovšem je si jí poupravit nadefinováním vlastním CSS[15] stylem.

Komponenta pro testování autentičnosti formulářů (Captcha):

Další z celé řady zajímavých komponent je ta pro testování formulářů pomocí Captcha testu. Tento test je velice jednoduchý a spočívá pouze v opsání stanového slova z obrázku do formulářového labelu. Tím se v aplikaci snaží zabránit přístupu robotů na místo reálných uživatelů. Díky tomu je možné zabránit například zahlcení aplikace automatickým opakováným vkládáním či obrana proti spamu v aplikaci. Tím se komponenta <com:TCaptcha> stáva velmi důležitou, zvláštně když jde o webovou aplikaci.

[15]CSS – Cascading style sheets. Kaskadové styly – jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML a XML.

5.1.7 Srovnání s ostatními frameworky

Prado framework je prvním velkým frameworkem, který začal využívat komponentového přístupu a událostmi řízeného programování. Od jiných frameworků se liší především použitím tříd a šablon. Tříd pak zastupují přirozenou logiku a šablony prezentační vrstvu. Naproti tomu ostatní frameworky jsou především zaměřené na MVC architekturu a snaží se pouze oddělit prezentační a aplikační vrstvu. Prado nám tak umožňuje maximálně využít propojení více komponent, kterých je k dispozici velké množství.

5.1.8 Zhodnocení

Prado framework se dostal do popředí díky výhře v soutěži Zend's PHP 5 Coding Contest. Celý tento framework se již od počátku vývoje značně odlišil od ostatních představitelů. Základy Prada je možné nalézt u Borlandu Delphi či ASP.NET. Jako první framework vůbec využívá komponentového přístupu a událostmi řízeného programování.

Hlavní myšlenkou tedy je využít maximální znuvupoužitelnosti při vývoje webové aplikace. Díky tomu můžeme využívat své vlastní komponenty s komponentami jiných programátorů. To vede k značné úspoře času a umožní nám tak vyvíjet aplikaci rychleji a snadněji. Kód by tak měl být vždy co nejjednodušší a pokud možno srozumitelný. Existují také pravidla, které určují jak vytvářet správnou komponentu či jí pouze vhodně poskládat již více existujících.

Vše je pak řízeno pomocí událostí. Díky tomu jsou jednotlivé aktivity uživatele zachyceny jako serverové události. Funkce či metody jsou následně automaticky volány při provedení každé události uživatele a snaží se na ní zareagovat vhodným způsobem.

Stejně jako u všech frameworků tak i u Prada platí, že vývoj aplikace je hlavně zaměřuje na implementaci aplikační logiky a prezenční vrstvy.

Velice kladně bych ohodnotil celkovou podporu tohoto frameworku. Jednak je možné najít na internetu dosti materiálů ke studiu a následnému vývoji aplikace mimo oficiální stránky a to i v českém jazyce. Hlavním zdrojem informací je pak ovšem oficiální stránka Prada frameworku. Zde je nalezení několik desítek komponent s příslušnou dokumentaci a rozborem kódu. Následně lze na stránkách nalézt manuál celého frameworku a tutoriály, které nás provedou při jeho prvním použití.

Další z celé řady výhod je možnost využití zabezpečení celé aplikace. Kdy je možné využít řadu metod jak ochránit aplikaci proti napadení. Prado framework má kompletní sadu cache technik. Ty se snaží co nejvíce urychlit požadavky na webovou aplikaci.

Za částečnou nevýhodu bychom možná mohli označit ne vždy zcela jasnou adresářovou strukturu a také větší hardwarové nároky. Problém může také nastat při samotném studiu, které je časově náročnější. To je zapříčiněno velikosti frameworku. Ovšem je zde strmá křivka učení takže doba studia se bude značně měnit s přibývajícími zkušenostmi.

Na závěr bych chtěl říct, že celý tento framework je vhodný nejen pro vývoj jednoduchých webových aplikací jako například blogy a podobně. Ale je také vhodný při tvorbě větších aplikací jako například podnikových systému. Zde se uplatní především dobrá srozumitelnost kódu čímž se skvěle hodí pro týmovou práci.

5.2 NETTE

5.2.1 Výhody, přednosti a vlastnosti

Znuvupoužitelnost - Díky podpoře znovupoužitelnosti je možné při vývoji kombinovat různé části různých aplikací, jenž byly navrhnuty v Nette frameworku. To má za následek úsporu času a snížení vytíženosti vývojáře.

Událostmi řízené programování - To umožňuje vývojáři soustředit se na interakci uživatele, neboť ostatní činnosti jsou zachyceny jako serverové události.

Podpora AJAXu - Dnes velmi moderní a jednoduché řešení, jak tvořit aplikace s využitím AJAXu bez nutnosti psát jednotlivé řádky kódu v JavaScriptu.

Ladící nástroje - Jedná se o velmi užitečný nástroj, který pomáhá odhalovat chyby aplikace. Je založen na knihovně /Debug/. Má dva režimy, ve kterých může detekovat chyby kódu. Prvním z nich je vývojový režim a ten má za úkol vypisovat výstup přímo na obrazovku pomocí chybového okna. Druhým režimem je produkční, v němž je chyba logována do souboru a případně také informace o chybě zaslána na e-mail. Tyto režimy jsou striktně odděleny z důvodu zabezpečení a volba režimu probíhá automaticky, ale v případě potřeby jej vývojář může nastavit ručně.

Cool URL **-** Nette framework vyniká na poli frameworku, také touto vlastností. Umožňuje nastavení tvaru URL adresy jako poslední věc při vývoji aplikace. Hezká URL adresa (neboli Cool URL) je důležitá pro SEO[16]. To má za následek postoupení na vyšší pozice ve vyhledáváčích. Zároveň jsou také adresy čitelnější a zapamatovatelnější pro uživatele stránek.

Zabezpečení webových aplikací **-** Díky neustálým útokům hackeru je nutné mít kvalitně zabezpečenou webovou aplikaci. Každá bezpečnostní díra zvyšuje riziko napadení. Právě použití Nette frameworku je tou nevhodnější volbou, jak se napadení bránit. Využívá několik metod, jak zabezpečit aplikaci.

První z metod je Context-aware escapin, která zabraňuje útoku Cross-site scripting (XSS). Díky technologii Context-aware escaping dochází k automatickému ošetření všech výstupů a tím je zabráněno podstrčení útočníkova kódu. Ten by měl za následek přístup k citlivým datům.

Druhou metodou je generování a oveřování autorizačního tokenu. Ten má za úkol eliminovat Cross-site reguest forgery (CSRF) útok. Tento útok nutí uživatele navštívit stránku, pomocí které skrytě napadne webovou aplikaci, kde je uživatel zrovna přihlášen. Tím hrozí změna obsahu aplikace, aniž by si toho uživatel povšimnul.

Třetí metodou je zcela automatické ošetřování vstupu dané aplikace. Automatické právě proto, že řada vývojářů opomíná důsledné ošetřování vstupů a zde vzniká obrovská bezpečnostní díra, pomocí které je možné využít útoku jako Control codes či URL attack. Tím vzniká snaha podstrčit uživateli webové aplikace škodlivý vstup. Což může mít za následek opět získání citlivých informací z databáze.

Poslední čtvrtá metoda zabezpečení spočívá ve správném nakonfigurování serveru a PHP. Tato nakonfigurování provádí Nette opět zcela automaticky. Vývojáři odpadá tedy starost, jak správně zabezpečit session. S tou souvisí cela řada útoků. Nejhorším z nich je zcizení session ID uživatele, a tím získání přístupu do dané webové aplikace. Poté již hacker může provádět v aplikaci jakékoliv kroky a úkony bez povšimnutí uživatele.

---

[16]SEO – Search Engine Optimization – jde o metodiku vytváření a upravování webových stránek takovým způsobem, aby jejich forma a obsah byly vhodné pro automatizované zpracování v internetových prohlížečích[23].

5.2.2 Požadavky na systém

Pro správnou činnost tohoto frameworku je nutné mít správně a funkčně nakonfigurovaný HTTP server. Pokud chceme můžeme pracovat také na webovém serveru, který musí podporovat PHP minimálně 5.2.0. Budeme-li chtít využít všech možností, vyšší zabezpečení a vyšší výkon, je doporučeno mít vždy aktuální verzi PHP a to tedy 5.3.2. To jsou tedy standardní požadavky. Nette ovšem rozlišuje dvě konfigurace ve kterých je schopný pracovat. První z nich je tak zvaná minimální. Tato konfigurace však může značně omezit použití a využití tohoto frameworku. Druhou konfigurací je tak zvaná doporučená, při které běží Nette se všemi vlastnostmi a rozšířeními. Před samotnou prací s frameworkem je doporučeno použít nástroj Requirements Checker. Tento nástroj je obsažen v distribuci Nette a má za úkol otestovat běhové prostředí serveru. Jako výstup toho nástroje je pak informační tabulka v prohlížeči, ve které informuje, zda (a do jaké míry) je možné framework používat. Instalace tohoto skriptu je jednoduchá. Prvním předpokladem je mít staženou distribuci Nette. V ní je nutné nalézt příslušný nástroj, který je standardně umístěný v /tools/Requirements-Checker. Z tohoto adresáře je následně nutné zkopírovat celý obsah na zvolený webový server. Poté už jen zbývá zadat do prohlížeče adresu skriptu a dojde k zobrazení výpisu. Po ověření a případném vyladění je z důvodu zabezpečení důležité Requirements Checker ze serveru odstranit.

5.2.3 Instalace

Instalace Nette frameworku je velmi jednoduchá. Prvním a základním krokem je nutnost stáhnout si Nette na svůj lokální disk. Zde může nastat jistý problém. Pokud se pozorně podíváme na stránky výrobce (http://nettephp.com/cs/download), zjistíme, že ke stažení je několik možných verzí. Aby nedošlo k omylům a pozdějším problémům jsou zde uvedeny hlavní rozdíly mezi těmito verzemi.

Základní rozdělení je takové, že se rozlišují dvě hlavní varianty:

1. ***Stabilní varianta***

   – u této varianty je zaručená plná spolehlivost. Neobsahuje ovšem nejnovější možnosti a nástroje, které tento Framework nabízí.

2. ***Vývojová varianta (Nightly Builds)***

   - s tou variantou máme k dispozici jeden z nejmodernější PHP frameworku vůbec, který nabízí značnou míru rozšíření od první

varianty. Tato varianta je bohužel nevhodná na nasazení při tvorbě projektů, neboť její stabilitu díky neustálému vývoji nelze zaručit.

Dále dochází ještě k jednomu dělení, které je také důležité brát v potaz. Každá z dvou výše uvedených variant se dělí dále na tři různé verze.

1) ***Verze pro PHP 5.2 – Jde o standardní verzi***

2) ***Verze pro PHP 5.2 s prefixy tříd***

   - prefixované názvy tříd jsou zde označeny pomocí písmene „N“ (NObject, NSession a podobně). Díky prefixu tříd je možné Nette Framework snadno a bezproblémově integrovat i s jinými PHP *knihovnami, které již názvy tříd jako Object či Html používají.*

3) ***Verze PHP 5.3 se jmennými prostory***

   - v této verzi dochází k plnému využití jmenných prostorů a dalších vylepšení vyskytujících se v PHP 5.3. Třídy jsou pojmenovany jako Nette\Object, Nette\Web\Session a podobně.

Instalace:

1) ***Vytvoření projektu***

Nejprve je nutné vytvořit si projekt a to tak, že vytvoříme adresář, který ponese název celého projektu. Struktura tedy bude mít tento tvar:

```
NazevProjektu/
```

2) ***Instalace Nette Frameworku***

Do vytvořeného adresáře si „nainstalujeme“ Nette Framework

1. Je nutné rozbalit archiv s frameworkem. Tento archiv jsme si stáhli na počátku instalace.
2. V rozbaleném archivu nalezneme adresář tools/Skeleton a jeho obsah zkopírujeme do adresáře NazevProjektu/. Pro připomenutí,

tento adresář jsme si vytvořili v předchozím kroku. Tímto dostáváme již základní strukturu aplikace.

3. Nyní do této základní struktury je nutné nakopírovat samotný Nette Framework. Ten je umístěn v námi rozbaleném adresáři pod názvem /Nette. Tento adresář nakopírujeme do adresáře NazevProjektu/libs/.

4. Právě byla dokončena instalace a je připravena k prvnímu spuštění.

3) Test instalace

1. Spusťe si váš webový server a prohlížeč nasměrujte do adresáře *NazevProjektu/document_root/*. V případě spuštění pomocí localhostu bude link vypadat následovně.
   http://localhost/NazevProjektu/document_root/

2. Pokud dopadla instalace úspěšně, v prohlížeči dojde k zobrazení uvítání a gratulaci ke spuštění.

5.2.4 Adresářová struktura

Po prvotní základní instalaci dostaneme základní adresářovou strukturu. Nette Framework lze samozřejmě používat také i bez této struktury a vytvořit si tak strukturu svou. Tímto je framework na adresářové struktuře **nezávislý**. Nicméně měnit tuto strukturu bych doporučil až zkušenějším uživatelům. Vnitřní adresářová struktura má tento formát.

```
NazevProjektu  /app/log
               /models
               /…
               /document_root/cs
               /images
               /…
               /libs/Nette
               /tests
```

Mezi hlavní části můžeme zařadit tři základní adresáře. Jak můžeme vidět výše, jde o adresáře nesoucí název /app, /document_root, /libs.

V tomto prvním adresáři, který nese název */app/*, je umístěná aplikační logika každé aplikace. Jde tedy o jeden z nejdůležitějších adresářů při vývoji webových aplikací. Díky tomu je také nutné dodržovat jisté zabezpečení, kdy skripty aplikační logiky jsou umístěny v odděleném adresáři, který je nedostupný z prohlížeče. Tento adresář se dále dělí na menší části.

NazevProjektu/app/presenters	- obsahuje jednotlivé vytvořené soubory presenteru (kontrolerů)
/models	- obsahuje jednotlivé modely dané aplikace
/templates	- obsahuje pohledy nebo jednotlivé šablony dané aplikace
/temp	- obsahuje dočasná data dané aplikace
/bootstrap.php	- jde o zaváděcí soubor, který nastaví prostředí aplikace a spustí jí
/config.ini	- obsahuje konfiguraci dané aplikace

V druhém adresáři */document_root/* můžeme nalézt například obrázky, různé javascripty dané aplikací a věci, co nejsou nutné k zabezpečení, jako například různé PHP skripty a to z důvodu toho, že tento adresář bude veřejně dostupný. Adresář by se měl tedy nacházet na webovém serveru, odkud by měl být přímo dostupný. Jedinou výjimku v umístění PHP souboru tvoří index.php. Do tohoto souboru budou směřovány všechny požadavky na danou webovou aplikaci. S tímto souborem také souvisí jedno ulehčení, které Nette nabízí. A to volba tvaru URL pro celou aplikaci, kdy tento tvar není nutné nastavovat na počátku vývoje pomocí pravidel mod_rewrite, jak je tomu zvykem, ale můžeme tuto činnost odložit až do okamžiku, kdy bude projekt hotov. Poté stačí jen určitá změna na daném místě (v index.php) a vše je vyřešeno. Výhodou může pro někoho také být volnost adresářové struktury. Dojde-li totiž k přesunu adresáře s aplikací logikou či knihovnami, postačí poté pouze otevřít soubor index.php a změnit v něm potřebnou cestu na novou.

NazevProjektu/document_root/css	- obsahuje soubory kaskádových stylů (CSS). Definují vzhled stránky
/images	- obsahuje obrázky, jenž jsou v aplikaci použity
/js	- obsahuje javascriptové skripty
/index.php	- směřují se zde požadavky dané aplikace

Třetím a také posledním adresářem ze základní struktury je adresář */libs/*. Do tohoto adresáře budou při vývoji aplikace umísťovány všechny různé knihovny a frameworky. Bez těchto rozšíření by byla aplikace velmi strohá a nebyla by plně využitelná.

```
NazevProjektu/libs/Nette                -obsahuje samotný Nette framework
```

5.2.5 Struktura a práce s frameworkem

Pro snadnější představu a orientaci, jak se s tímto PHP frameworkem pracuje, bude vše ukázáno na názorných příkladech. Hlavní pro nás bude, jakým způsobem se implementuje návrhový vzor MVC a které náležitosti je nutno dodržovat.

Návrh Modelu:

Jak jste si mohli již přečíst na počátku práce, model je zodpovědný za přístup a modifikaci dat v tabulce databáze. Z toho nám tedy vyplývá, že každá jednotlivá tabulka databáze by měla mít svůj vlastní model, který jí zastupuje. Následně jsou pak data z modelu předána presenteru a ten jej pošle šabloně. Ta nakonec tato data zobrazí.

Model v Nette frameworku nemá pevně danou formu a tak dává vývojáři jistou volnost při jeho realizaci. U modelů v Nette bývá jako databázová vrstva využíván databázovy layer Dibi. Za databázovým layerem Dibi stojí opět lidé okolo Davida Grudla a tak je spojení logické. Nicméně toto spojení není podmínečné a tudíž je možné zvolit i jinou variantu.

Než začneme se samotným návrhem je nutné se vůbec připojit k databázi. Otevřeme si soubor config.ini a vyplníme údaje nutné pro připojení k databázi. Nyní už jen doplníme do bootstrap.php nasledující a připojení je kompletní.

```
dibi::connect(Environment::getConfig('database'));
```

Dále postoupíme k samotnému návrhu modelu.

V adresáři /NazevProjektu/app/models/ vytvoříme dvě nové třídy. První třída bude zastupovat záznam z databáze. Druhá třída bude obstarávat vrácení záznamu z databáze, a to

ve formátu dané třídy. Při návrhu těchto tříd využijeme návrhového vzoru Active Record. Nyní se vrátíme zpět k třídám. Vytvoříme první třídu s názvem Todo. Ta bude obsahovat metody jako save() nebo delete(). Jakmile je první třída hotová, přichází na řadu druhá, kterou pojmenujeme TodoManager. Ta nám zajistí, že potřebná data přejdou k presenteru. A to formátu třídy Todo nebo jako pole objektů Todo.

Návrh Šablony:

Šablona je další ze tří částí MVC architektury. Ač název šablona není standardní, jedná se o část, která je obvykle pojmenována View. Ta má za úkol poskytnout rozhraní pro prezentovaní dat uživateli. Stejně tak slouží i k zadávání požadavků v podobě formulářů a následnému uložení dat.

Šablony v Nette jsou pojmenovány podle akce, kterou porvádí a celý název je doplněn příslušnou příponou. Název tedy může vypadat následovně: zobraz.phtml. Jméno šablony by mělo začít malým písmenem naopak jméno presenteru velkým, o tom ale až později. Tato pravidla pojmenování jsou konvence, které jsou důležité pro spravný chod frameworku. Samotná šablona je umístěna v adresáři presenteru, ke kterému patří.

Důležité je si při vývoji zapamatovat, že šablona jedné konkrétní akce patří vždy k jednomu konkrétnímu presenteru. Je-li potřeba, umožňuje Nette tak zvané dvoufázové renderování šablon. Při použití dvoufázového renderování šablon dochází k tomu, že ke každé akci již není použita jedna šablona, ale dvě. První nese název šablona akce a druhá šablona layoutu. První šablona má za úkol zobrazení a obsah stránky. Tato šablona je uložená v adresáři presenteru. Druhá šablona je ta, v níž je uložen layout. Šablonu layoutu lze snadno poznat podle názvu, neboť vždy začíná znakem „@“. Tato šablona je uložená přímo v adresáři /templates/, neboť nepřísluší k žádnému konkrétnímu presenteru. Důvod je jednoduchý, tato šablona je společná pro několik různých akcí. Tím se liší od první z šablon. Pro vývojáře začátečníky je dobré používat výchozí soubor této šablony. Ten je možné opět naleznou v adresáři /templates/ pod názvem @layout.phtml. Výchozí layout pak není nutné nikterak nastavovat a Nette framework tento soubor automaticky nalezne a v případě jeho existence ho použije. Takže u této šablony postačí upravit pouze kód dle svých potřeb a je možné jí využívat.

Návrh Presenteru:

Presenter je poslední ze tří částí MVC architektury. U Nette Frameworku došlo k nahrazení standardního Controlleru právě Presenterem. Stejně jako u MVC návrhového vzoru je i zde Presenter stěžejní a nejdůležitější částí. Má za úkol předávat požadavky a akce Modelu. Je tedy jakousi ústřední jednotkou, která má za úkol postarat se o celkové provázání funkčnosti aplikace. Rozdíl však u Nette nastává v počtu Pressenteru. Jejich počet může být různý. Při vývoji je však dobré myslet na to, že čím méně jich bude, tím lépe.

Implementace presenteru není až tak složitá. Ovšem čím více presenteru vývojář naimplementuje, tím budou další následné kvalitnější. Než se přesuneme k samotné implementaci, připomenu, že název presenteru začíná vždy velkým písmenem.

Nyní můžeme začít s tvorbou samotnou. Nejprve je nutné najít v adresářové struktuře adresář /presenters. Do něj pak vytvoříme soubor NazevprojektuPresenter.php. Do toho souboru následně vložíme tento kód.

```
<?php

final class NazevprojektuPresenter extends BasePresenter { public function
actionZobraz(){}public function renderZobraz(){}
```

V tomto kódu vidíme hned několik věcí. Nejprve jsme vytvořili finální třídu, díky které vzniká náš Presenter. Ten zároveň dědí od BasePresenteru. V těle třídy nalezneme funkci, díky které půjde zobrazovat obsah. Funkce jsou dvě, neboť první - actionZobraz - pracuje s modelem a druhá - renderZobraz - pracuje s pohledem. Nyní je základní Presenter hotov.

Nastavení práce s databázovým serverem:

Pokud chce aplikace pracovat s nějakými daty, vzniká potřeba tato data někam ukládat. Proto přichází na řadu krok, kdy je nutné navrhnout a vytvořit databázi. Databáze bude realizována pomocí MySQL. Toto však není zavazující pravidlo a je možné použít jinou z databází spolupracující s Nette frameworkem.

Před samotným návrhem databáze je nejprve vhodné se zamyslet a zvážit, co je třeba, aby naše databáze vůbec uměla ukládat.

Nyní přichází samotný návrh. Ten probíhá pomocí SQL příkazů, a tak si zde vývojář nevystačí pouze s PHP jazykem. Základním krokem je vytvoření databáze spolu s tabulkou.

```
SET NAMES utf8; - nastavení způsobu kódování

CREATE DATABASE - `nazevprojektu`; - vytvoření databáze

USE - `nazevprojektu`;

DROP TABLE IF EXISTS - `zkouska`;   - odstranění tabulky s daným názvem, pokud již existuje

CREATE TABLE - `zkouska` (- vytvoření tabulky- zde jsou nutné jednotlivé atributy)
 ENGINE = InnoDB DEFAULT CHARSET = utf8;

INSERT INTO - `zkouska` - vložení testovacích dat pro ověření funkčnosti databáze
   VALUES (- nastavení hodnot daných atributu)
```

5.2.6 Zajímavé vlastnosti a funkce

Nette Framework nabízí celou řadu komponent a doplňků. Díky nim můžeme tento framework doplnit o celou řadu zajímavých funkcí, které se moc často u ostatních frameworků nevyskytují. Standardně nejsou tyto rozšíření součástí frameworku, a tak je nutné jej dodatečně dohledat a poté nainstalovat. Většina všech komponent, doplňků a také různých Plutonu, je vyvíjena komunitou okolo tohoto frameworku a jsou dostupné na stránkách výrobce. Jelikož doplňků je celá řada, není možné představit zcela všechny, a tak se pokusím vybrat alespoň pár z nich, jejichž funkce podrobněji popíšu.

Komponenta pro platbu kartou přes 3-D Secure (GP webpay):

Komponenta *WebPay 0.3.* představuje moderní řešení realizace platby přes bránu 3-D Secure. Umožní uživateli, využívající aplikaci, realizovat okamžitou bezhotovostní platbu, například za zprostředkované služby. WebPay 0.3 se ovládá pomocí událostí a má za úkol zapouzdřit odchozí a příchozí požadavek.

Událostmi řízená komponenta ovládá životní cyklus platby. Ta je složena z menších, na sebe navazujících událostí.

1. *Událost na vytvoření požadavku*
   -zde dochází k volání události WebPay::$onCreate

2. *Událost při odpovědi od banky*
   -zde dochází k volání události WebPay::$onResponse. Tato událost je vyvolaná při pozitivní odpovědi z banky. Jinými slovy, dojde-li k zaplacení požadavku.

3. *Událost při chybové odpovědi od banky*
   -zde dochází k volání události WebPay::$onError. Tato událost je vyvolaná při záporné odpovědi z banky. Jinými slovy, dojde-li k neuhrazení požadavku.

Naproti tomu, aby vůbec došlo k vyvolání události, je nutné uživateli samotnou platbu umožnit. O to se stará WebPay control. Ten dovolí realizovat platbu pomocí tlačítka nebo formuláře.

Tlačítko

Využije se v případě, kdy postačí uživatelovo potvrzení (pomocí buttonu). Ten pouze klikne na daný button a automaticky dojde k platbě. Tato metoda je vhodná pouze tehdy pokud není třeba znát žádné vstupy, jako například jméno, email a podobně.

Formulář

Napojení na formulář je druhý způsob jak realizovat platbu. Tato metoda se využije všude tam, kde bude zapotřebí zadat vstupní hodnoty, jako například jméno, email a podobně. Formulář můžeme nalézt v adresáři */Forms/* a k napojení využijeme událost na Form::$onSubmit:. Zde je dobře patrné, jak snadné může být napojení na */Forms/*.

Samotná instalace této komponenty vyžaduje dva kroky. Prvním krokem je uzavření smlouvy s bankou pro povolení platby kartou na základě, které Vám budou přiděleny patřičné údaje nutné ke správné funkci. A poté, co bude první krok vyřešen, nezbývá nic

jiného, než vložení samotné komponenty WebPay do adresáře aplikace, přesněji tedy adresáře /Components.

Komponenta pro validaci formulářů (Live Form Validation):

Jednou z dalších zajímavých volitelných komponent je i ta pro validaci formulářů. Zajímavá je z důvodu odlišností od standardních validací, nacházejících se ve frameworcích, tak i od té v Nette. Tato komponenta validuje formulář již za běhu při vyplňování (control po controlu), nikoliv až při samotném odeslání. Díky tomu je možné ihned informovat uživatele o chybném vyplnění.

Informování o chybném vyplnění je zobrazeno vedle formuláře, pomocí varovného zobrazení. Validační pravidla je možné upravovat, aby odpovídala požadavkům dané aplikace. Pro tyto nastavení je nutné pracovat se souborem *LiveClientScript.php.* Tímto je možné ovlivnit chování live validace.

Samotná instalace není nikterak těžká. Nejprve je nutné komponentu stáhnout ze stránek distributora a následně rozbalit. Dostaneme tak dva soubory. První z nich (*LiveClientScript.php)* má za úkol renderovat validaci a druhý *(LiveFormValidation.j*s) definovat pomocné javascriptové funkce. Na závěr je nutné nastavit proměnnou ClientScript na novou instanci třídy LiveClientScript a do šablon includovat soubor LiveValidation.js. Proměnná ClientScript se nachází v adresáři Renderer. Tímto je základní instalace dokončena.

5.2.7 Srovnání s ostatními frameworky

Nette framework je ryze českým představitelem. Na české scéně je suverénní jedničkou.

Za konkurencí může tento framework zaostávat díky své neúplné dokumentaci. Nicméně je možné na většinu nejasnosti nalézt odpovědi ve fóru. Tam je soustředěna celá komunita. Výhodou je ovšem široká možnost zabezpečení a velký výkon. Díky tomu je tento framework jeden z nejrychlejších vůbec. Co se týče historie, patří Nette k těm mladším představitelům. Nicméně pomalu, ale jistě se dostává do popředí a je tedy dobrou volbou zejména pro tuzemského vývojáře.

5.2.8 Zhodnocení

Nette framework vyniká svou jednoduchostí a efektivitou použitelnosti. Studium práce v tomto frameworku je snadná a příjemná. Díky velmi kvalitně zpracovaným tutoriálům je čas strávený studiem dosti zkrácen a tak se zde může porovnávat z nejsilnějšími frameworky. To znamená, že křivka učení má u tohoto frameworku strmý vzestup a tím se může stát vývojář pracující s Nette velmi produktivní.

Adresářová struktura je přehledná a logická. Jelikož není nikterak striktně omezená danými pravidly, je možné jí upravovat dle potřeb. Kód frameworku je jasný a ve velké míře je rozšířen řadou komentářů, které pomáhají v jeho orientaci. Také nevyžaduje omezující pravidla a konvence pro psaní programu.

Nette, také vyniká jedním z nejlepších ladicích nástrojů v poli frameworku. To umožní rychle najít chyby i méně zkušenějším vývojářům, popřípadě odhalit nepatrné překlepy těm zkušenějším.

Jako další přednost bych viděl velmi propracované zabezpečení, kdy tento framework dokáže odolat mnohým z útoků a pokusu o nabourání. To vše jen díky použití moderních a revolučních technologií. Zabezpečením lze například ošetřit i nahlížení vývojáře na kód pomocí IP adresy. Právě proto je Nette velmi oblíbený i na důvěrné weby. Jeden příklad za všechny, je oficiální stránka současného prezidenta Václava Klause, jenž je vyvíjená v Nette.

Dalším s celé řady předností je jeho rychlost a kombinovatelnost. Tento framework patří k těm nerychlejším a rozdíl je mnohdy znatelný. Kombinovatelností je zde umožněno spojení s jinými PHP frameworky, popřípadě také spojení například s javascriptovými frameworky

Předností je tedy celá řada, ovšem je nutné, se také zaměřit na nedostatky. Jistou drobnou nevýhodou oproti velkým frameworkům může být to, že za vývojem stojí komunita v čele s panem Davidem Grádlem, nikoliv však silná firma. Z počátku byl vývoj díky tomu značně omezen. Nicméně v poslední době jde rychle kupředu a snaží se tak tuto nevýhodu eliminovat. Stále však bych nedoporučil nasadit tento framework na velké projekty.

Je vhodný spíše pro webové aplikace středního typu. A stále ještě může být pro někoho jistou slabinou ne zcela kompletní dokumentace. Musím ovšem říci, že dokumentace se postupně rozrůstá a je jen otázkou času kdy bude zcela kompletní.

6 Závěr

Cílem a záměrem této bakalářská práce bylo seznámení s tématem PHP frameworků. Práce byla tvořena s ohledem na nulové znalosti tématu možného čtenáře. Před samotným studiem bych ovšem doporučoval proniknout nejprve do tajů skriptovacího jazyka PHP.

Jedním z nejdůležitějších a ne přímo jednoduchých kroků je výběr správného frameworku. Důležité před samotným výběrem je nutnost ujasnit si požadavky a vlastnosti, jaké budou vyžadovány pro danou aplikaci. Dobré je také se zamyslet nad nasazením frameworku. Obecně můžu říci, že vhodné je nasadit framework vždy při vývoji aplikace či systému. Nicméně existují výjimky, pro které bych nedoporučoval nasazovat tuto softwarovou strukturu. Pokud jde o drobnou aplikaci s minimem funkcí či jednoduché stránky se statickým obsahem, stojí jistě za zvážení nasazení frameworku. A to především pokud nemáte předchozí znalosti s jeho prací. Studium dokumentace a tutoriálů by v tomto případě mohlo zabrat více času než implementace samotné aplikace a nevyužili bychom tak žádnou z výhod. Jinak bych doporučoval využívat těchto moderních řešení, a to nejenom v jazyce PHP.

Postupem času zjistíme kolik věcí jste si zjednodušili a vylepšili díky jejich použití frameworků.

Co se týče práce s frameworky, musím říct, že je opravdu příjemná. Je to možná také proto, že jsem měl vždy k dispozici kvalitní dokumentaci, díky které byly vyřešeny téměř veškeré všechny problémy. Pokud jsem již narazil na nějakou drobnost, která nebyla zmíněna v dokumentaci, využil jsem fórum, kde jsem vznesl dotaz s tímto problémem a řešil jej s lidmi z komunity. Rád bych také zde vyzdvihnul přednášku Davida Grudla o Nette Frameworku, kterou jsem měl možnost navštívit. Právě tato přednáška mne velmi zaujala a přivedla na myšlenku začít hlouběji se studiem tohoto frameworku.

Pevně věřím, že Vám tato práce poskytla dostatečný přehled o vlastnostech, možnostech využití a důvodech, proč právě při vývoji webových aplikací využít PHP Frameworků.

Seznam použití literatury

[1] PHP [online]. 2009 [cit. 21-11-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/PHP/>.

[2] Common Gateway Interface [online]. 2009 [cit. 21-11-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/Common_Gateway_Interface/>.

[3] Unicode [online]. 2009 [cit. 21-11-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/UNICODE>.

[4] Framework [online]. 2009 [cit. 28-11-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/Framework/>.

[5] Web application framework [online]. 2009 [cit. 4-12-2009].
Dostupný z WWW: <http://en.wikipedia.org/wiki/Web_application_framework/>.

[6] JUnit [online]. 2009 [cit. 4-12-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/JUnit/>.

[7] JQuery [online]. 2009 [cit. 4-12-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/JQuery/>.

[8] Úvod do architektury MVC [online]. 2009 [cit. 7-12-2009].
Dostupný z WWW: <http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc/>.

[9] Model-view-controller [online]. 2009 [cit. 13-12-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/Model-view-controller/>.

[10] AJAX [online]. 2009 [cit. 16-12-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/AJAX/>.

[11] CakePHP [online]. 2009 [cit. 17-12-2009].
Dostupný z WWW: <http://cs.wikipedia.org/wiki/CakePHP/>.

[12] All things CakePHP :: The Cookbook [online]. 2009 [cit. 17-12-2009].
Dostupný z WWW: <http://book.cakephp.org/>.

[13] PRADO PHP framework [online]. 2009 [cit. 19-12-2009].
Dostupný z WWW: <http://www.xisc.com/>.

[14] Velký test PHP framework: Fuse, Prado, Qcodo [online]. 2009 [cit. 19-12-2009]. Dostupný z WWW: <http://www.root.cz/clanky/velky-test-php-frameworku-fuse-prado-a-qcodo/>.

[15] JSON [online]. 2009 [cit. 19-12-2009]. Dostupný z WWW: <http://www.json.org/json-cz.html/>.

[16] Simple Object Access Protocol [online]. 2009 [cit. 19-12-2009]. Dostupný z WWW:<http://cs.wikipedia.org/wiki/Simple_Object_Access_Protocol/>.

[17] Nette Framework [online]. 2009 [cit. 20-12-2009]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Nette_Framework/>.

[18] Symfony | SYMBIO [online]. 2009 [cit. 26-12-2009]. Dostupný z WWW: <http://www.symbio.cz/slovnik/symfony.html/>.

[19] CodeIgniter [online]. 2009 [cit. 26-12-2009]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/CodeIgniter/>.

[20] Zend Framework [online]. 2009 [cit. 27-12-2009]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Zend_Framework/>.

[21] Velký test PHP framework: Zend,Nette,PHP a RoR [online]. 2009 [cit. 27-12-2009]. Dostupný z WWW: <http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/>.

[22] PHP framework: Prado [online]. 2010 [cit. 24-1-2010]. Dostupný z WWW: <http://www.czechdesign.cz/blogs/aichi/php-framework-prado/>.

[22] Search Engine Optimization [online]. 2010 [cit. 27-2-2010]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Search_Engine_Optimization/>.

Seznam příloh

Obsah přiloženého CD

Adresář	Obsah
/dokument	- Bakalářská práce v elektronické podobě
/dokumentace	- Uživatelská a programátorská příručka
/abstrakt	- Abstrakt v češtině a abstrakt v angličtině
/komponenta_CZ	- Implementace české verze komponenty pro Nette framework
/component_AJ	- Implementace anglické verze komponenty pro Nette framework
/support	- Instalační soubory podpor